On QoS-aware Scheduling of Data Stream Applications over Fog Computing Infrastructures

Valeria Cardellini, Vincenzo Grassi, Francesco Lo Presti, Matteo Nardelli
Department of Civil Engineering and Computer Science Engineering
University of Rome “Tor Vergata”, Italy

Fifth International Workshop on Management of Cloud and Smart City Systems 2015

Introduction

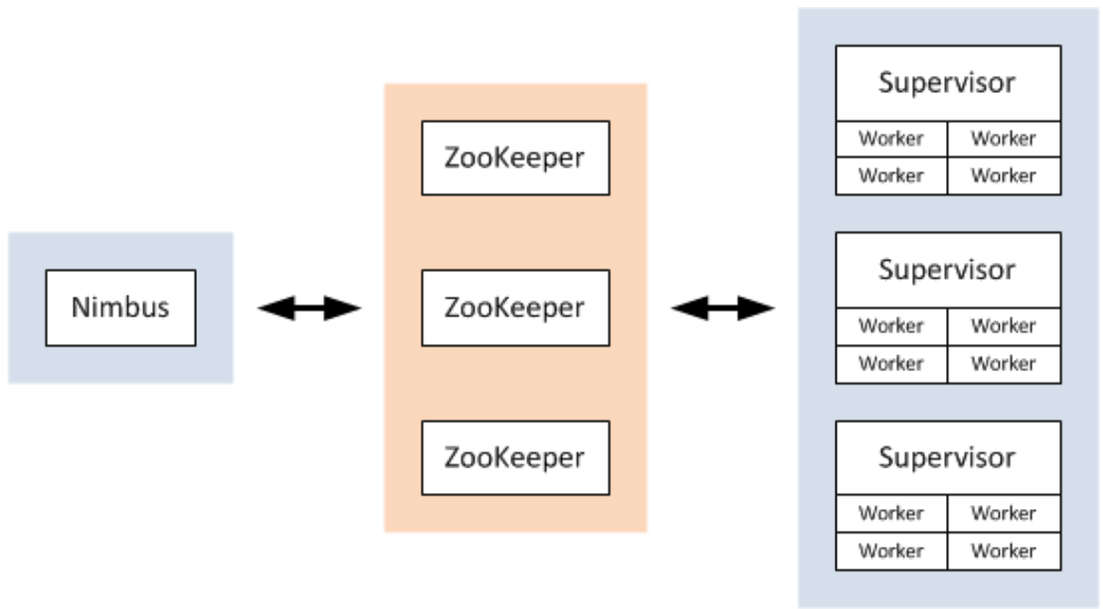
- Motivation
  - Data stream processing (DSP) in fog computing
  - Extract the useful information from raw data to improve urban services
- Challenges
  - Network and system heterogeneity
  - Dynamic geographic distribution
  - Non-negligible network latencies

Data Stream Processing in Storm

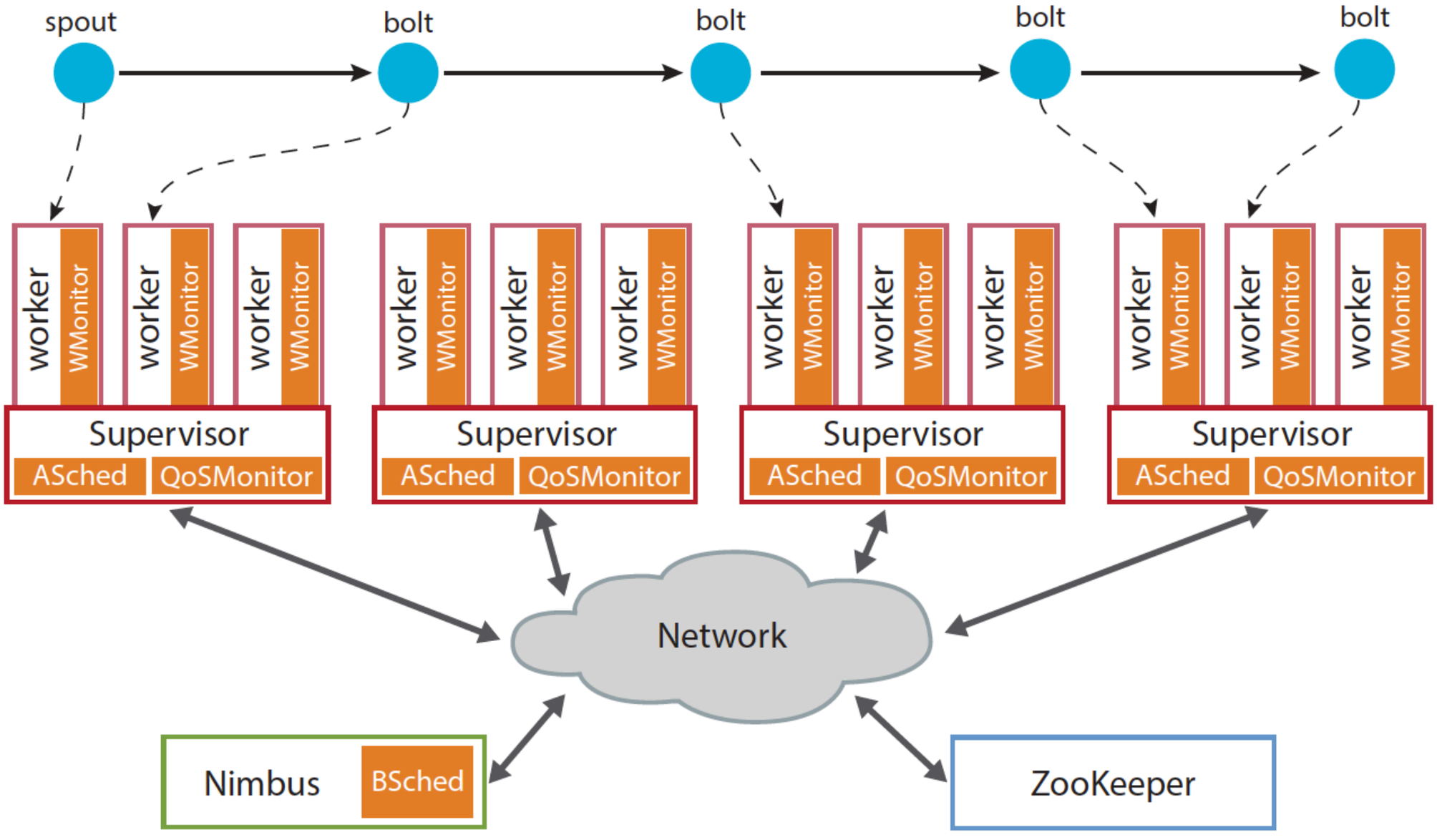
- An open source, real-time, and scalable DSP system maintained by the Apache Software Foundation
- Three types of entities to execute a topology
  - Task - An instance of an application operator
  - Executor - One or more tasks related to the same operator
  - Worker Process - Runs one or more executors of the same topology

Data Stream Processing in Storm

- Worker node - Generic computational resource
- Zookeeper - Shared memory service for managing configuration information and enabling distributed coordination
- Nimbus - Centralized component in charge of coordinating the topology execution
- Supervisor - Starts or terminates worker processes on the basis of the Nimbus assignments



Distributed Scheduling in Storm



Monitoring Compoents

- QoSMonitor
  - Estimates the network latency with respect to the other system nodes and monitors node availability and its resources utilization
- WorkerMonitor
  - Computes the data rate exchanged among the application components
- Nodes will share their informations with each others

AdaptiveScheduler

- Executes the distributed QoS-aware scheduling algorithm on every worker node
- Reassign executors to improve the application performance
- MAPE
  - Monitor - Identifies the set of local executors that could be moved
  - Analyze - Determines if the movable candidate will be effectively moved to another position
  - Plan - Determines a worker node that will execute the candidate executor
  - Execute

BootstrapScheduler

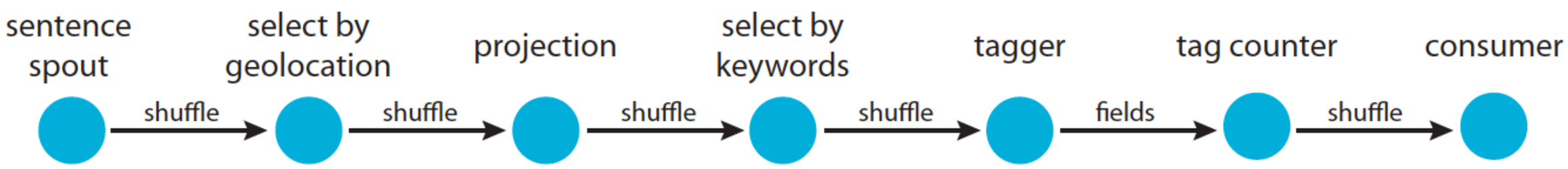
- A centralized scheduler, which defines the initial assignment of the application, monitors its execution, and restarts failed executors

Experimental Results

- DSP system is deployed on a network with not negligible latency and is subject to changes in the QoS of the nodes
- Focus only on network latency and node utilization
- Testbed
  - Apache Storm 0.9.3
  - 8 Worker nodes
  - 2 Further nodes for Nimbus and ZooKeeper.
  - Using “netem” to emulate wide-area network latencies
- Two sets of applications
  - simple topology with different requirements
  - well known applications

Adaptation Capabilities

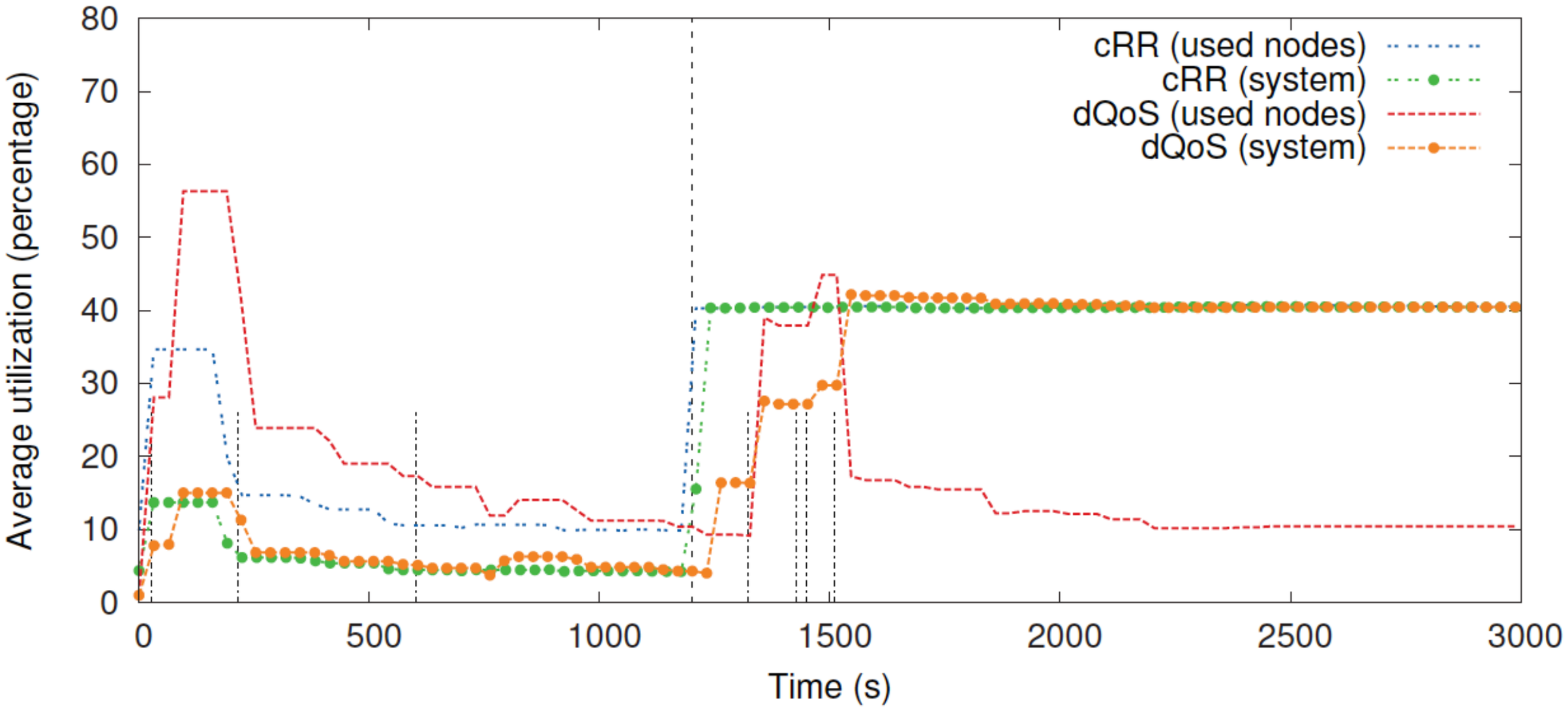
- Compare the two schedulers when the load experienced by the worker nodes changes during the application execute
- Use a simple application which tags and counts sentences produced by a data source



Base Line

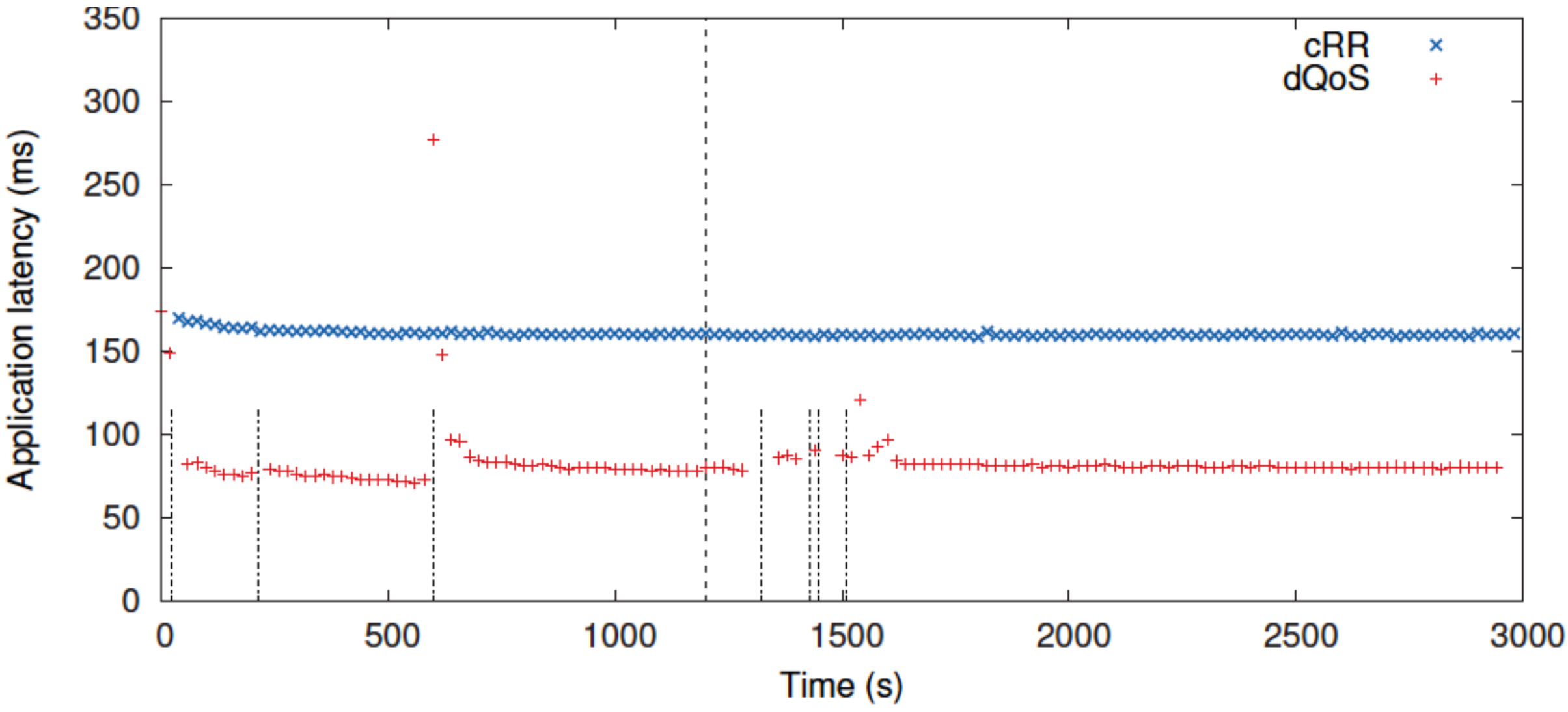
- A single executor for each operator
- After 1200 s, they artificially increase the load on a subset of three nodes using the Linux tool “stress”

Base Line



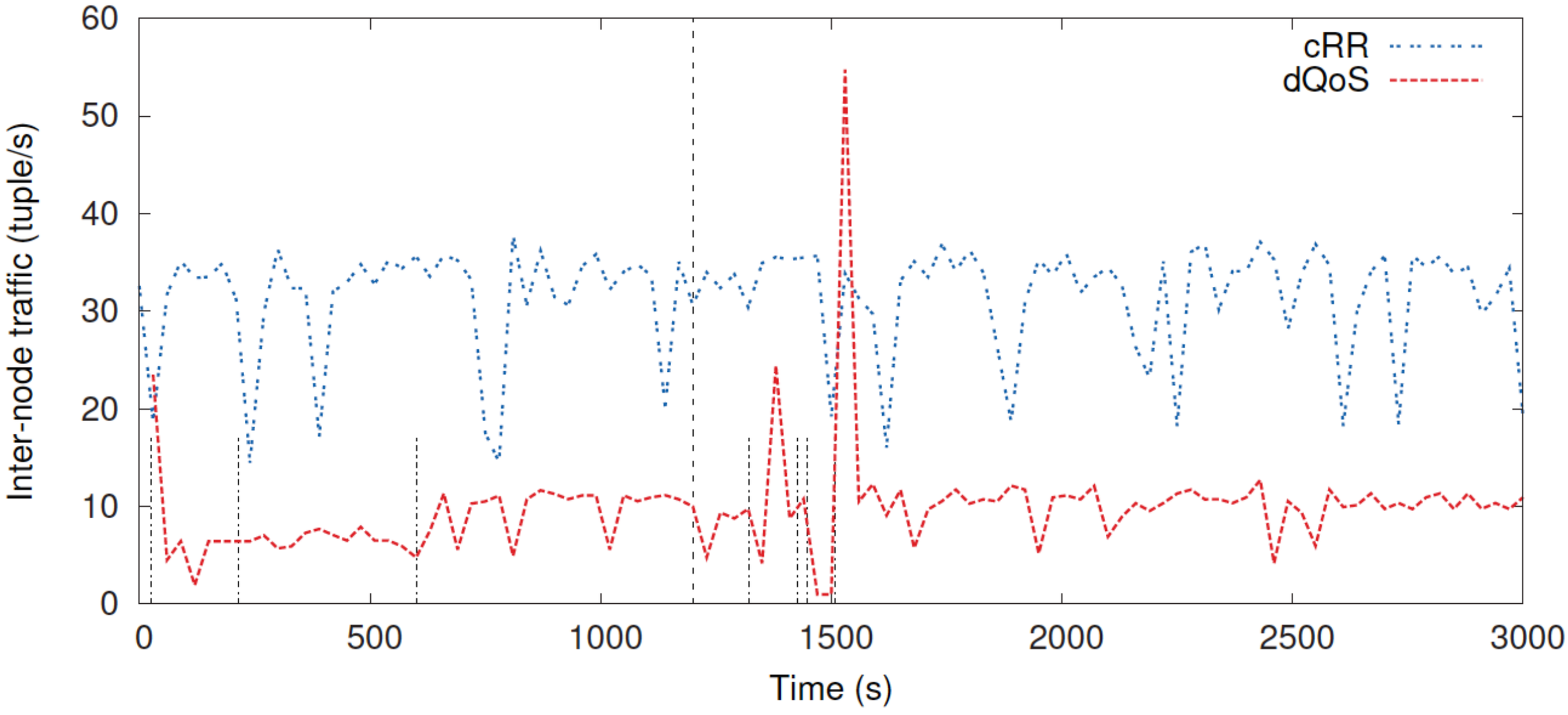
(a) Average node utilization

Base Line



(b) Application latency

Base Line



(c) Inter-node traffic

Heavy Application

- They modify the operators of the tag-and-count topology in order to waste some CPU time
- Load stress event is launched at 2450 s

Heavy Application

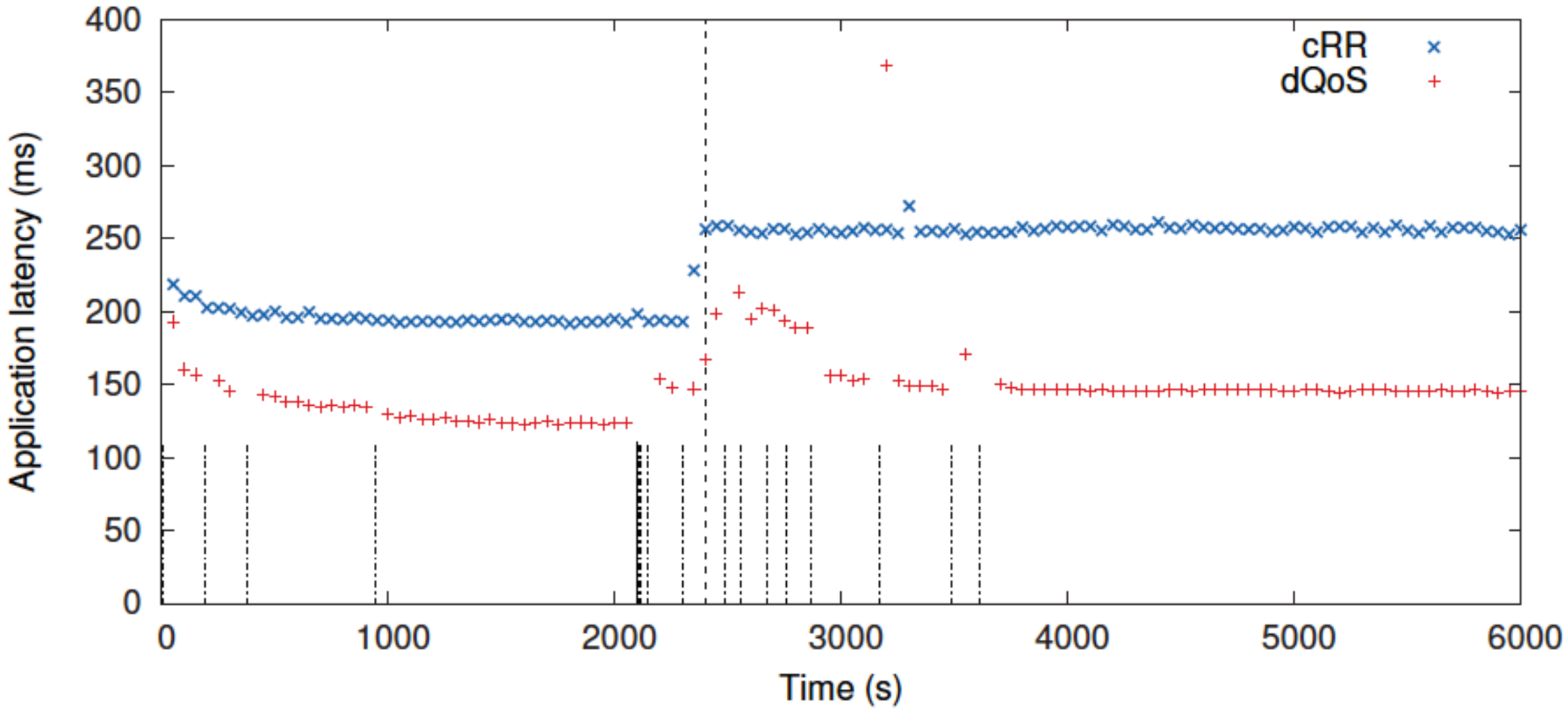
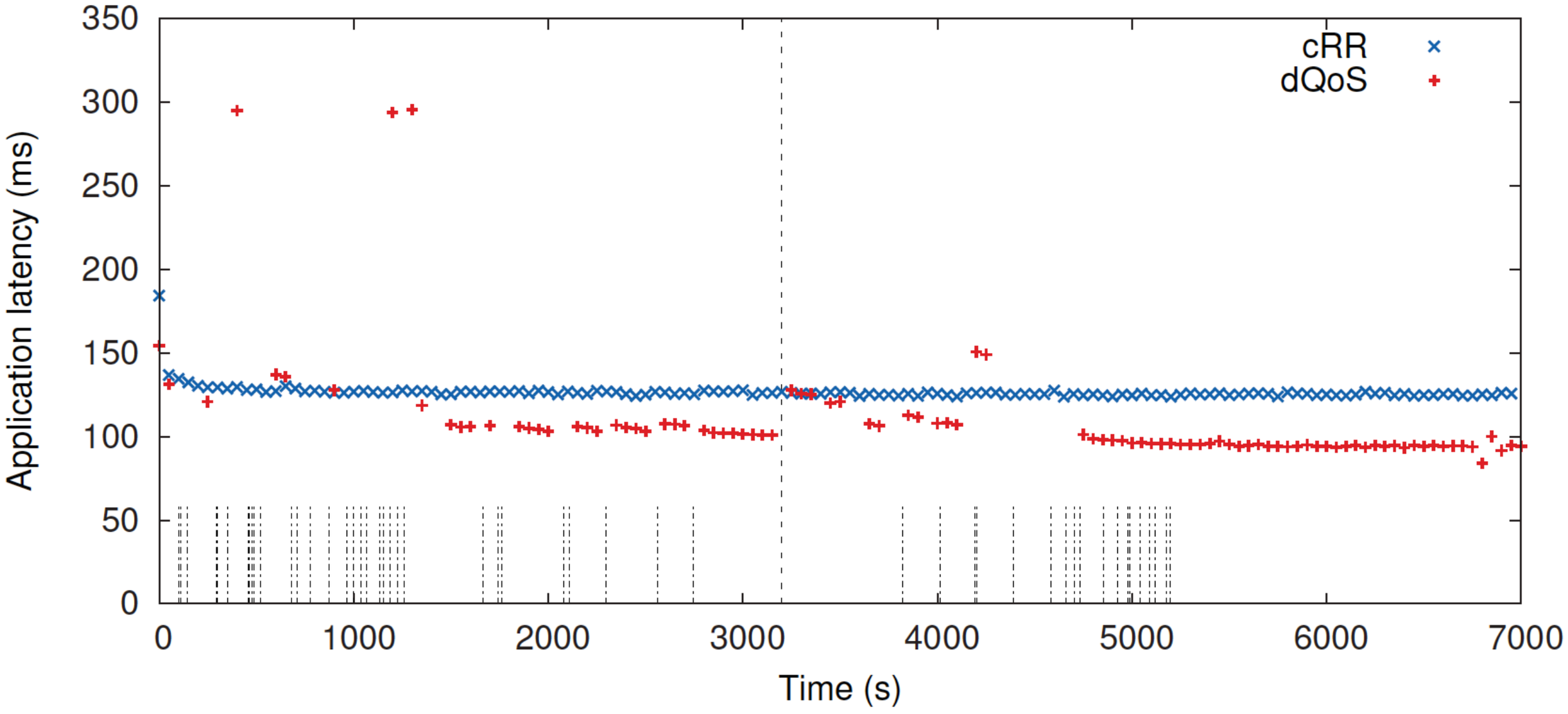


Fig. 4: Average latency of the "heavy" version of the tag-and-count topology when the nodes' utilization changes

Replicated Operators

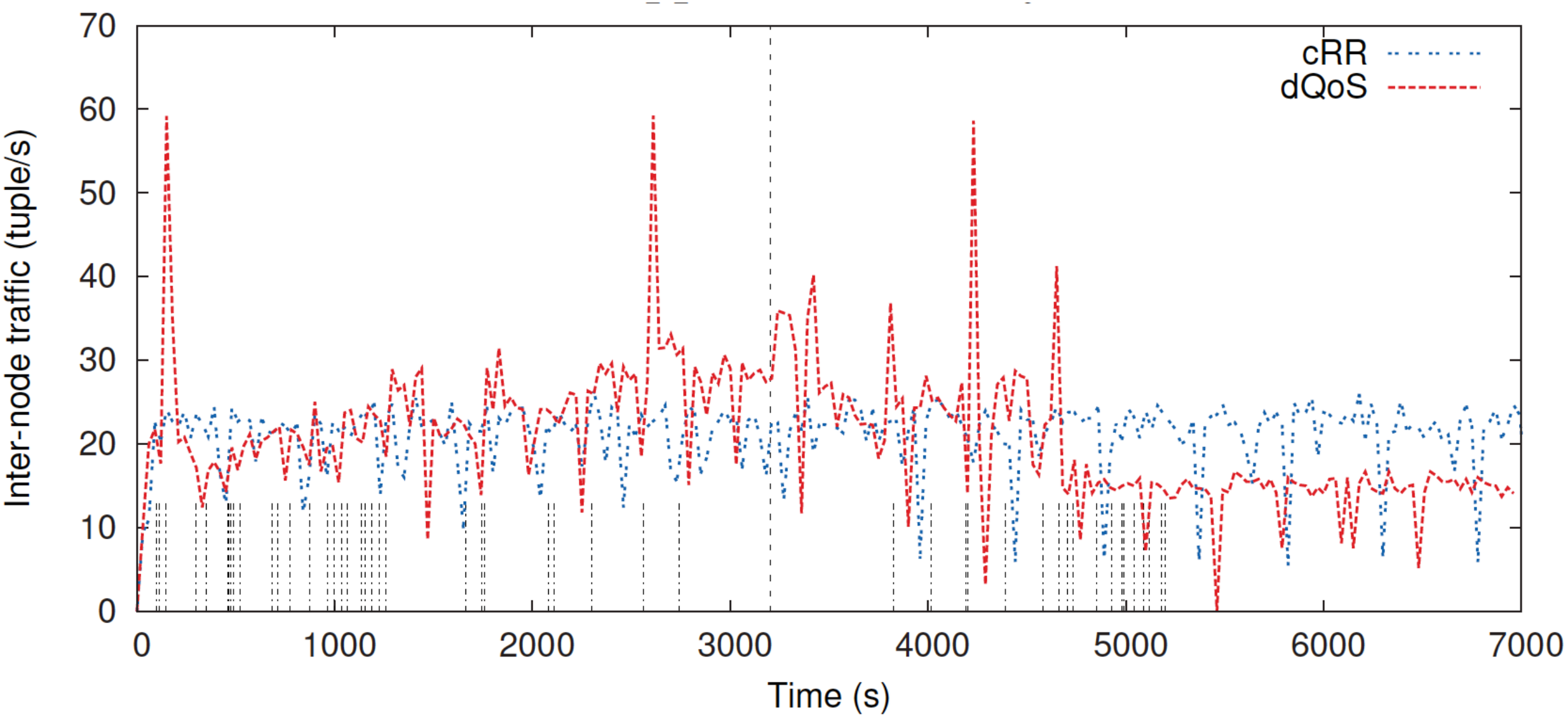
- Two executors are assigned to each operator
- The stress event is launched at 3200 s

Replicated Operators



(a) Application latency

Replicated Operators

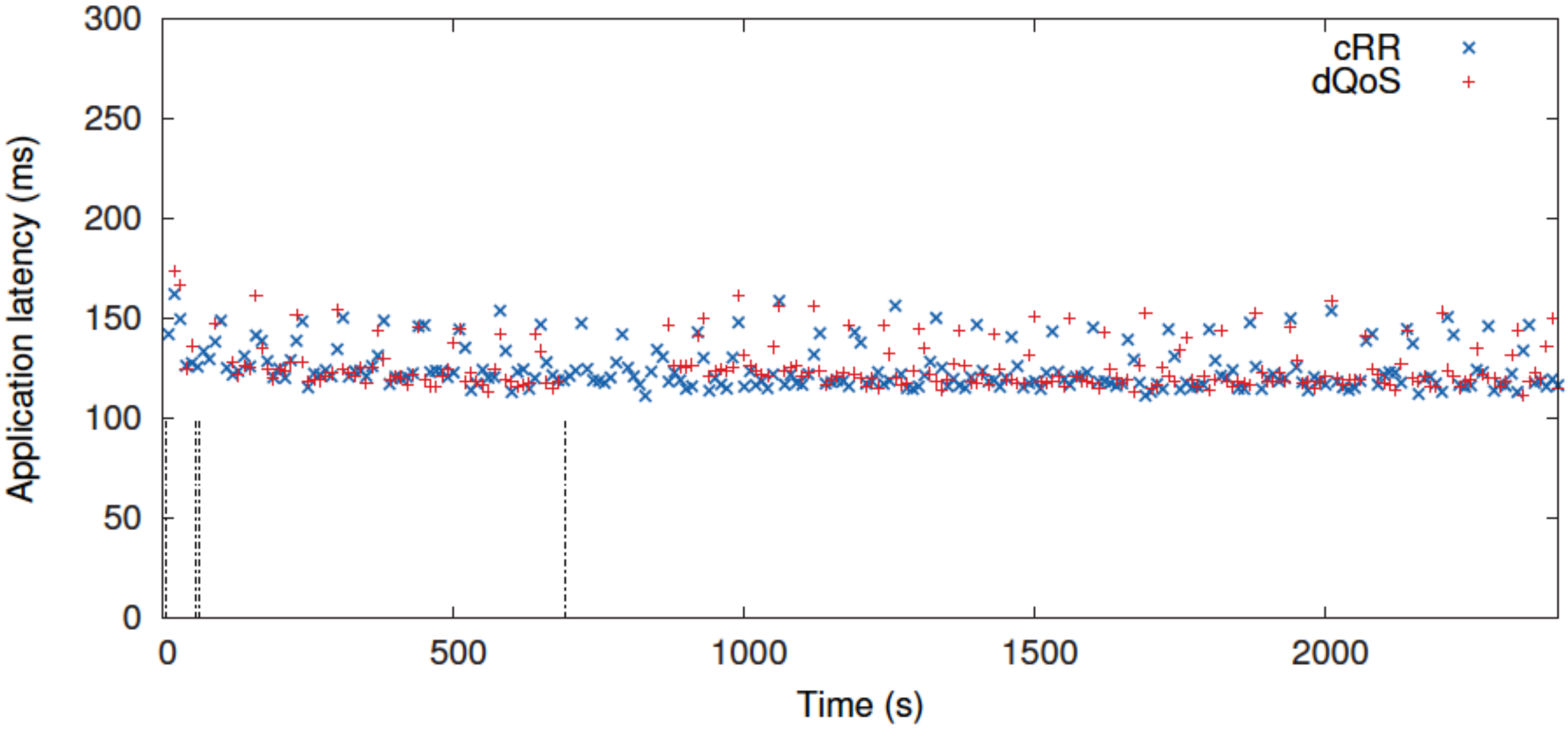


(b) Inter-node traffic

Well-known Applications (Word Count)

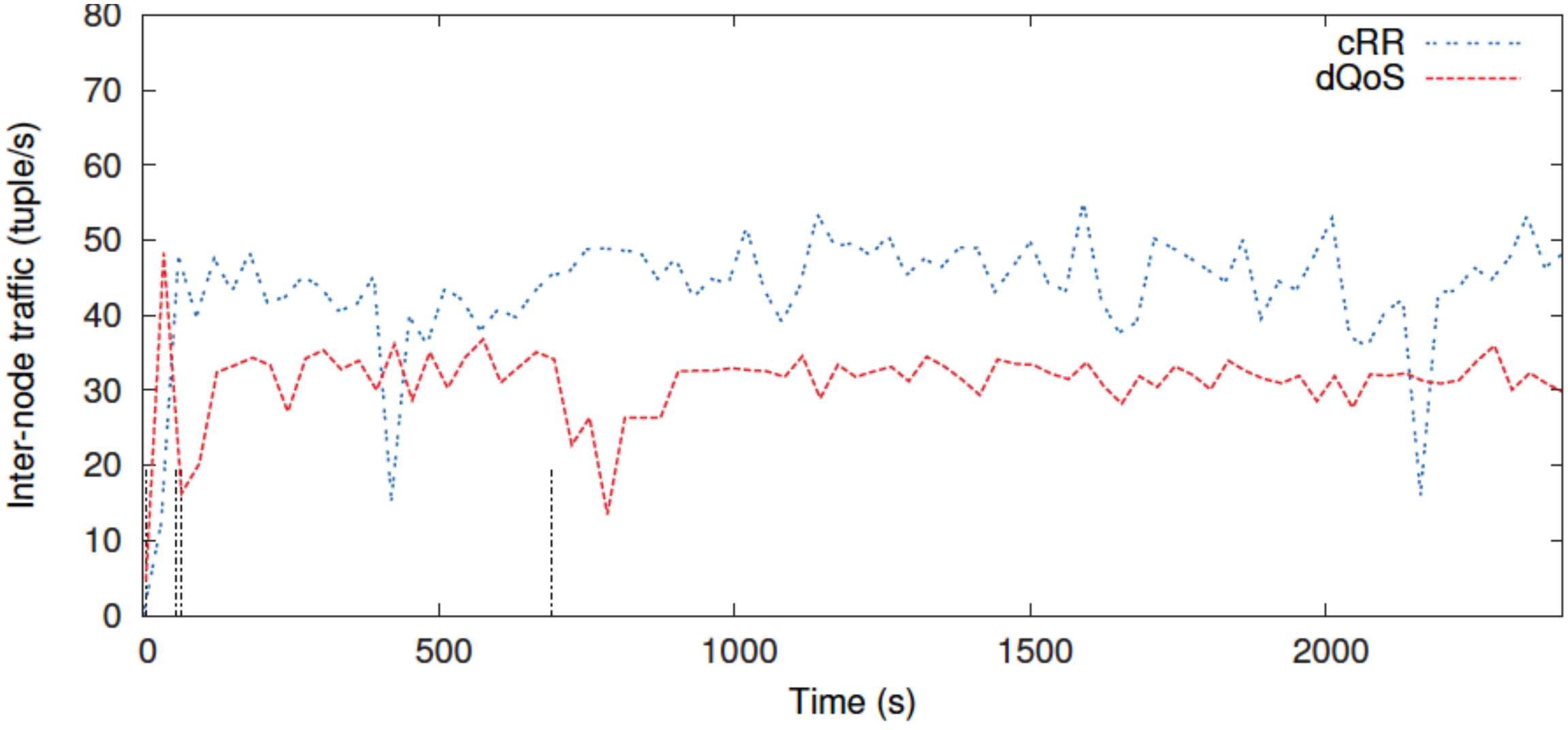
- Composed by a sequence of a source, two operators, and a consumer.
- The first operator splits the sentence into words and feeds the next one, which counts the occurrence of each word; each update of the counters is notified to the consumer
- Assign two executors to the source and three executors to each other operator

Well-known Applications (Word Count)



(a) Application latency

Well-known Applications (Word Count)



(b) Inter-node traffic

Well-known Applications (Log Processing)

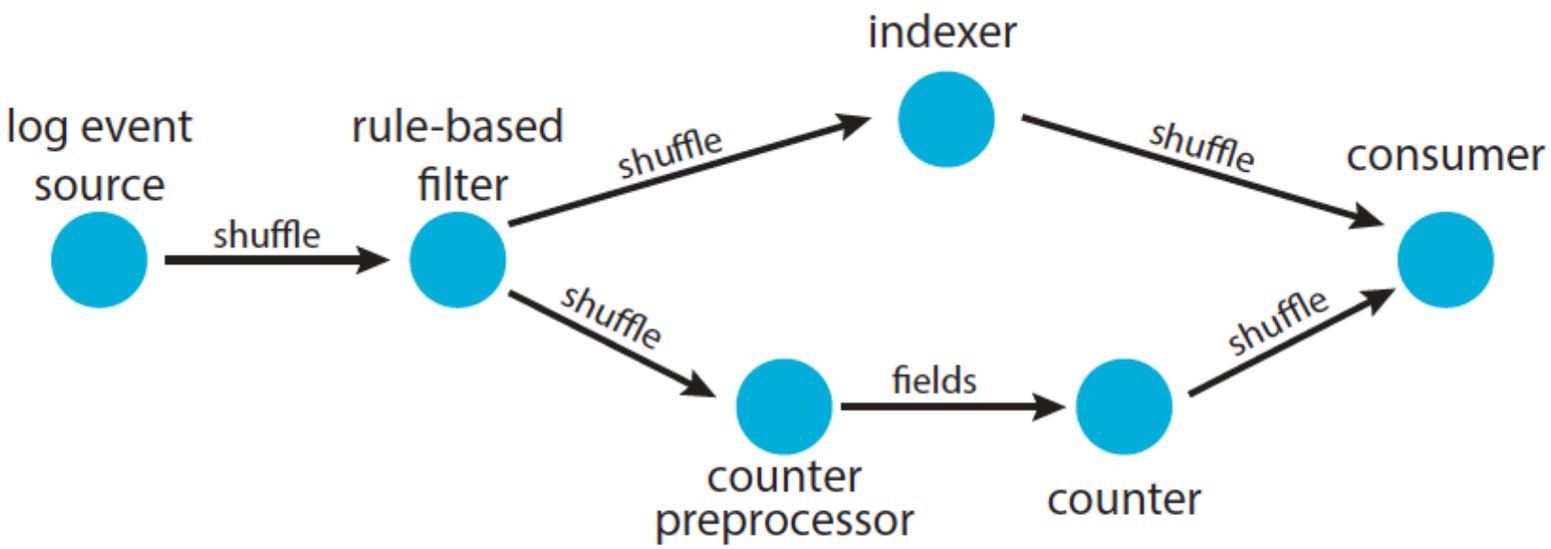
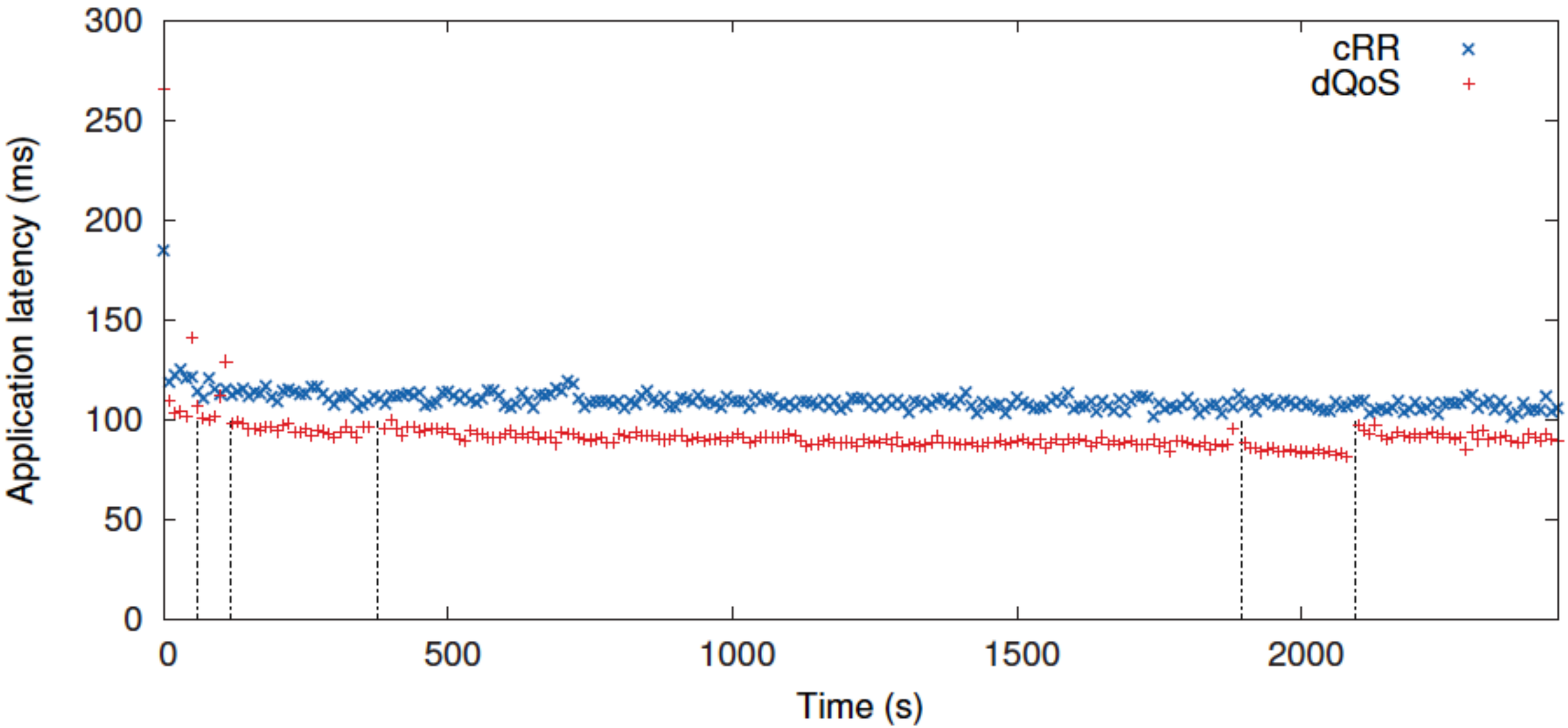


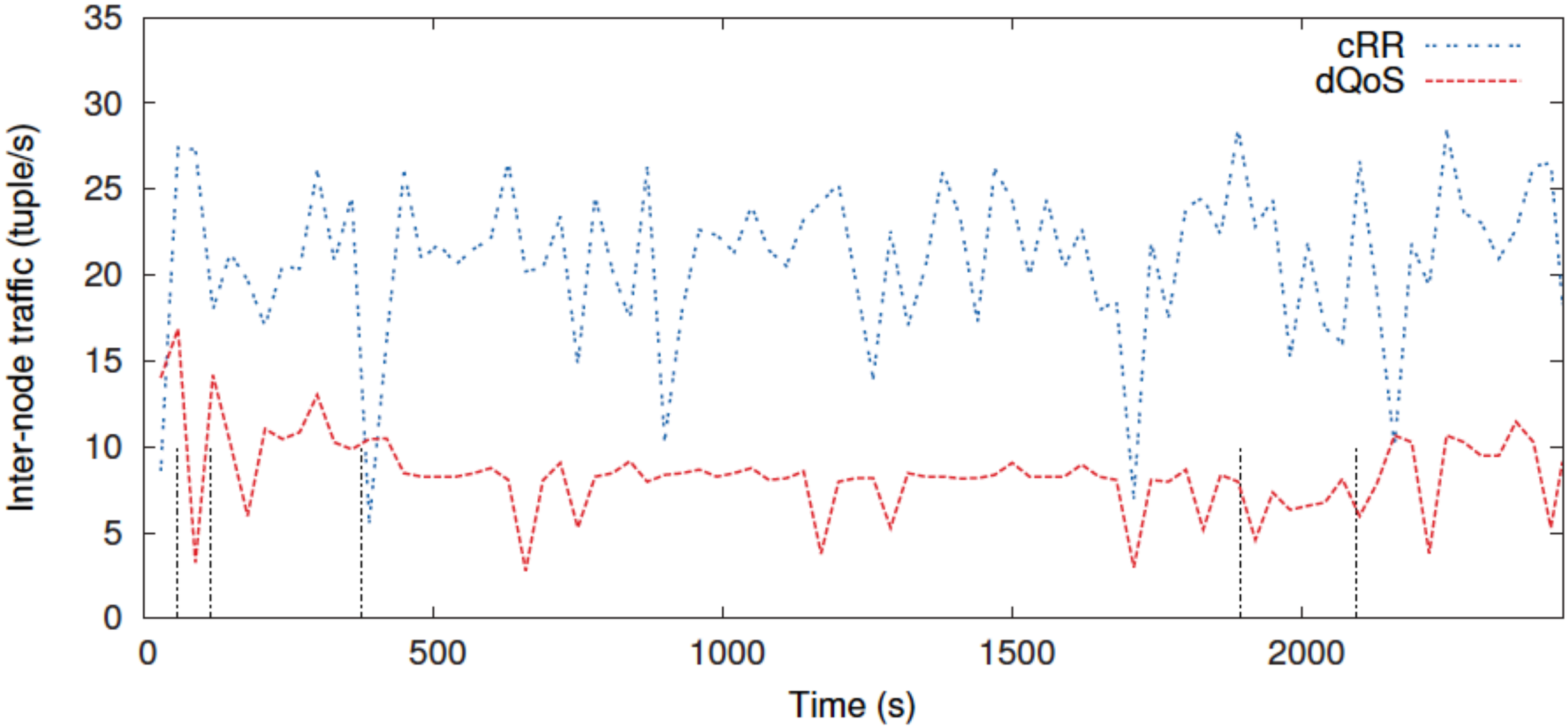
Fig. 7: Log Processing topology

Well-known Applications (Log Processing)



(a) Application latency

Well-known Applications (Log Processing)



(b) Inter-node traffic

Conclusion

- They have used two sets of applications evaluated the distributed QoS-aware scheduler for DSP systems based on Storm with
- The results show that their scheduler outperforms the Storm default one, improving the application performance
- Each placement decision is taken in a independent manner, for complex topologies involving many operators, it can determine some instability that affects negatively the application availability