The Chorus in the Chaos: When Big Data Platforms Meet Small IoT Devices
Son Nam Nguyen1, Teng Wang1, Ranjan Dahal1, Bin Zhao2, and Bo Sheng1
1Department of Computer Science, University of Massachusetts Boston
2School of Computer Science and Technology, Nanjing Normal University

Motivation: Big Data processing on IoT
- IoT devices participate in the computation rather than being merely the data source
- Evolved hardware capability
- Reduce data transfer traffic
- Reduce server load

Problems
- IoT devices rely on Wi-Fi connections
  - Low and unreliable network bandwidth
  - Shuffling data takes a long time
- Serious self-interferences
  - Big data tasks tend to finish in waves and shuffle their data around the same time.
  - Packet scheduling without knowledge of the big data jobs

Solution (A Token Based Packet Scheduler)
- Centralized control to avoid self-interferences
- Only the node granted with the token can transmit packets in a time window
- Assign the token and adjust the window size
- Run-time job execution info (app layer)
- Link qualities (data link layer)

Packet Scheduling Algorithm (master node)
- Derive the Estimated Transfer Time (ETT) for each node
  - The token is granted to the node (i) with the largest ETT. Its window size ($W_i$) is determined based on the prediction of the data generation.
  - $W_i = \arg\max_t \{ ETT_i(t) > ETT_j(t), \text{for any } j \neq i \}$
- Among multiple jobs, the job that is close to the end of map phase is given a higher priority.

Packet Control Module (slave nodes)
- Enforce the packet schedule by capturing all outgoing shuffling packets into a buffer.
- The buffered packets are sent only during the scheduled time window.
- Handle lost control messages and transmission overtime

Evaluation
- We implement our solution on Raspberry Pi 3 with Hadoop Yarn 2.7.2

Experiments
- Testbed: a cluster of 9 Raspberry Pis (1 master and 8 slave nodes). All nodes are connected in a WiFi ad-hoc network.
- Workload: Hadoop benchmark jobs
  - Sort: fixed and large intermediate data size
  - WordCount: various and small intermediate data size

Results
- Figure 2 shows the WiFi traffic of our solution when executing the same job as in Fig.1.
- Figure 3 compares the shuffling time of our solution and native Hadoop when executing Sort and WordCount jobs. The improvements on the shuffling time range from 12.7% to 30.7%.

Contact
Son Nam Nguyen
University of Massachusetts Boston
Email: Sonnam.nguyen001@umb.edu

Source Code
https://github.com/bboycoi/RPi-Hadoop