Network Quality of Service in Docker Containers

Ayush Dusia, Yang Yang, and Michela Taufer
University of Delaware

Objective: Improve the network QoS to ensure high performance of applications hosted in Docker containers

Limitations

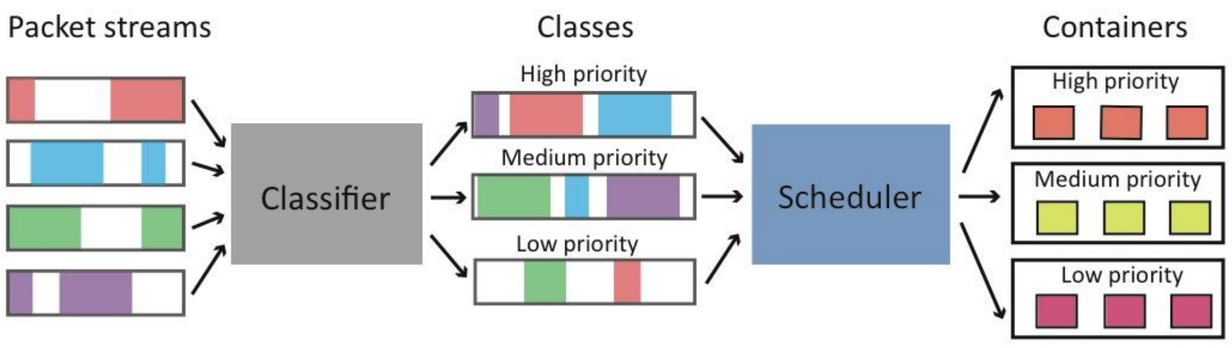
- Limited options are available in Docker to configure the network usage of containers, and the default "Best effort" is used
- Time-sensitive or bandwidth-intensive application results in poor or unacceptable performance with the default configuration
- No preferential treatment provided to the network traffic of high priority and critical applications hosted in containers
- Required level of service to containers cannot be ensured without expanding or over provisioning the available network bandwidth

Our solution: Extend Docker to include network priority and bandwidth throttling schemes that ensure QoS

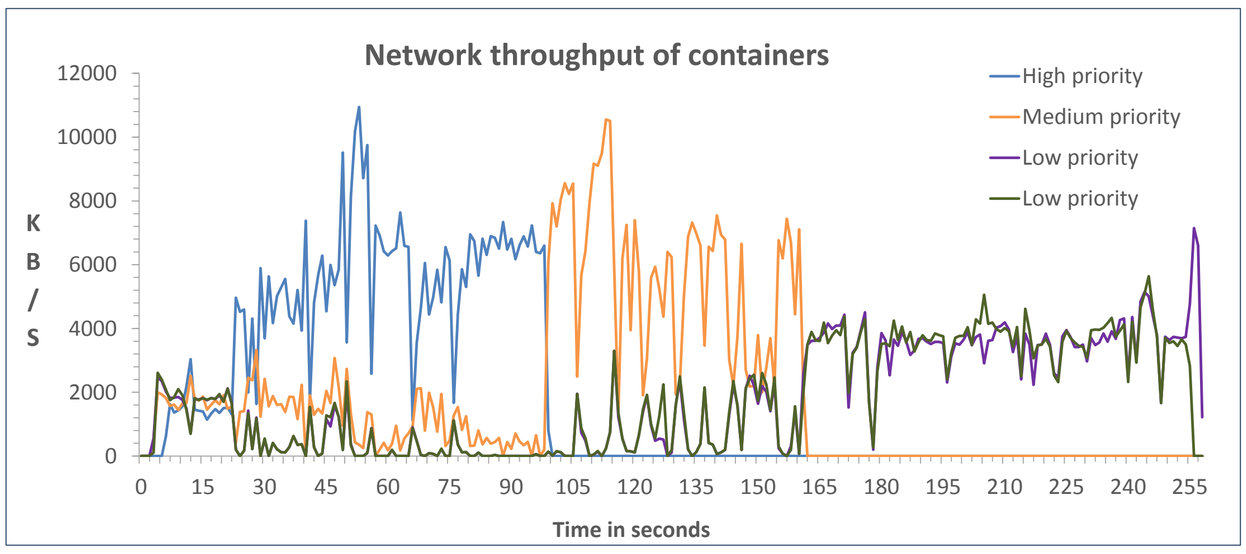
- A priority scheme is implemented in Docker by adding a **packet classifier** and **scheduler**, which enables preferential delivery service and priority to network traffic of user-sensitive applications, such as multimedia, or high bandwidth applications
- Network traffic of containers is **shaped** and **throttled** to minimize bandwidth congestion, optimize performance and improve latency

Priority scheme

- A packet classifier and scheduler is added on top of the *docker0* Ethernet bridge by configuring the Linux Traffic Control (TC) PRIO *qdisc*
- Network packets are classified and scheduled by the configured *qdisc*
- Three classes (priorities) are configured – High, Medium and Low
- Packets are filtered and classified using the IP address of containers
- Dequeuing of packets from the three classes ensures scheduling and hence a priority scheme for the network traffic



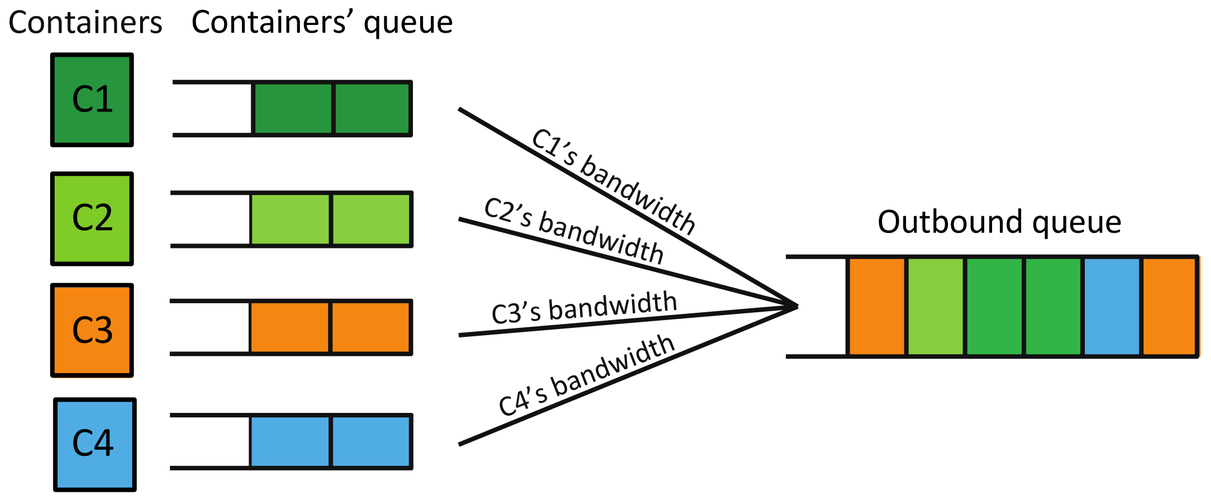
- Containers with high, default (medium) and low priorities are started to simultaneously download identical files of size 450 MB
- Network throughput is monitored for each container



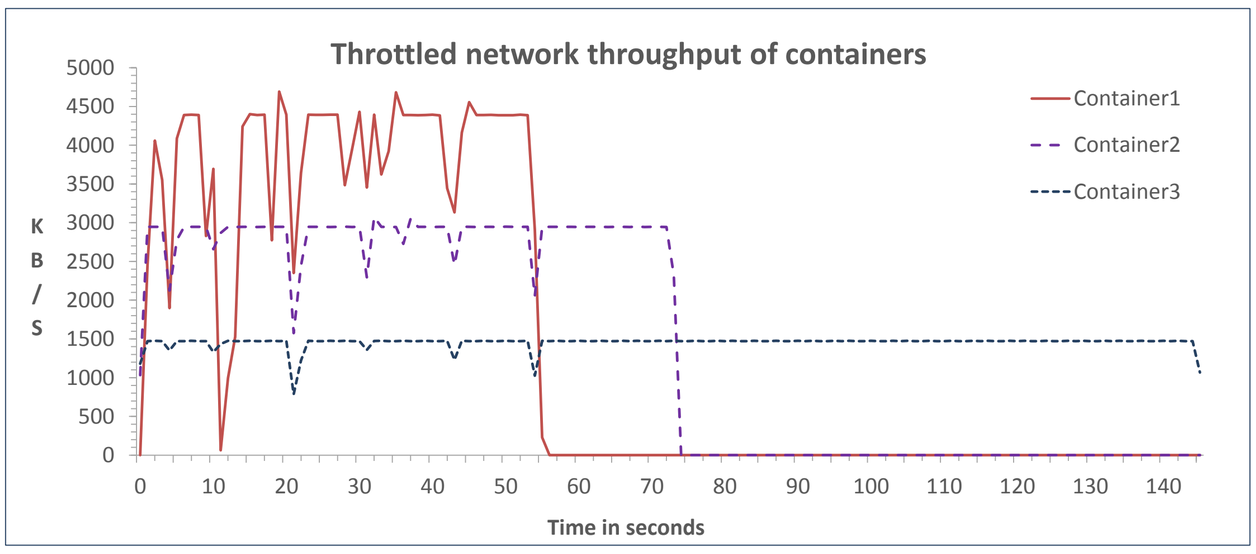
Higher share of the available total network throughput is observed for the higher priority container

Network bandwidth throttling

- Network traffic of containers is shaped and throttled to assigned limit
- Linux TC Token Bucket Filtering (TBF) *qdisc* is configured on the veth interface of each container to shape and throttle the network traffic
- TBF is a classless *qdisc* so containers can be assigned different limits
- Limit and burst size are configured according to required throughput



- Containers with restricted bandwidth – 4.5 MB/s, 3 MB/s and 1.5 MB/s are started to simultaneously upload identical files of size 200 MB



The network throughput of containers are throttled at different limits that are assigned at startup

Impacts and conclusions

- QoS in Docker improves the user experience and reduces the operation cost by allowing for efficient use of the existing resources
- Our extension allows Docker to appropriately throttle network traffic of containers to achieve the desired bandwidth sharing across all containers
- User-sensitive or critical applications, such as real-time multimedia can be hosted in higher priority containers to obtain preferential services
- We test our implementation with Docker v1.5 installed on Ubuntu 14.04 LTS
- The source code of our implementation is available on GitHub at - https://github.com/adusia/docker
- **Acknowledgment**: The authors acknowledge Dr. Seetharami R. Seelam for his advice and for giving us access to the IBM cloud, SuperVessel