CLOUD NATIVE CON
Europe 2017

KubeCon
A CNCF EVENT

Sched.net: A network aware Kubernetes scheduler

Berlin, March 2017

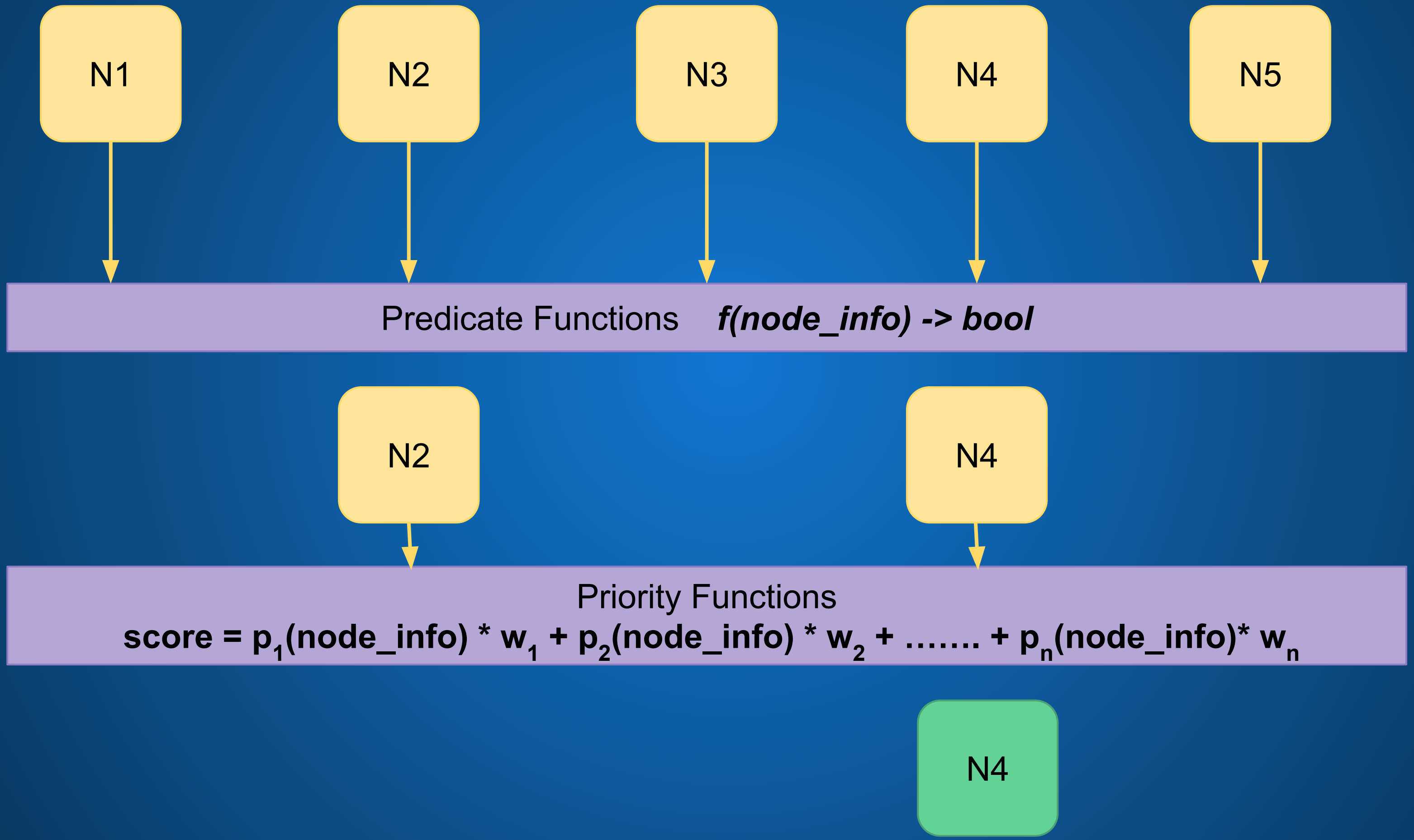
Akash Gangil
@aakashgangil
Sr. Software Engineer, Uhana

Salvatore Orlando
@taturiello
Staff Engineer, VMware

Agenda

- Default Scheduler
- Sched.Net
- OVN
- Setup topology
- Demo
- Results
- Questions

Default Scheduler



Predicate Functions ***f(node_info) -> bool***

Priority Functions

$$\textbf{score} = p_1(\text{node\_info}) * w_1 + p_2(\text{node\_info}) * w_2 + \ldots\ldots + p_n(\text{node\_info}) * w_n$$

Default Predicates

- PodFitsHostPort: Verify ports are available on the node.

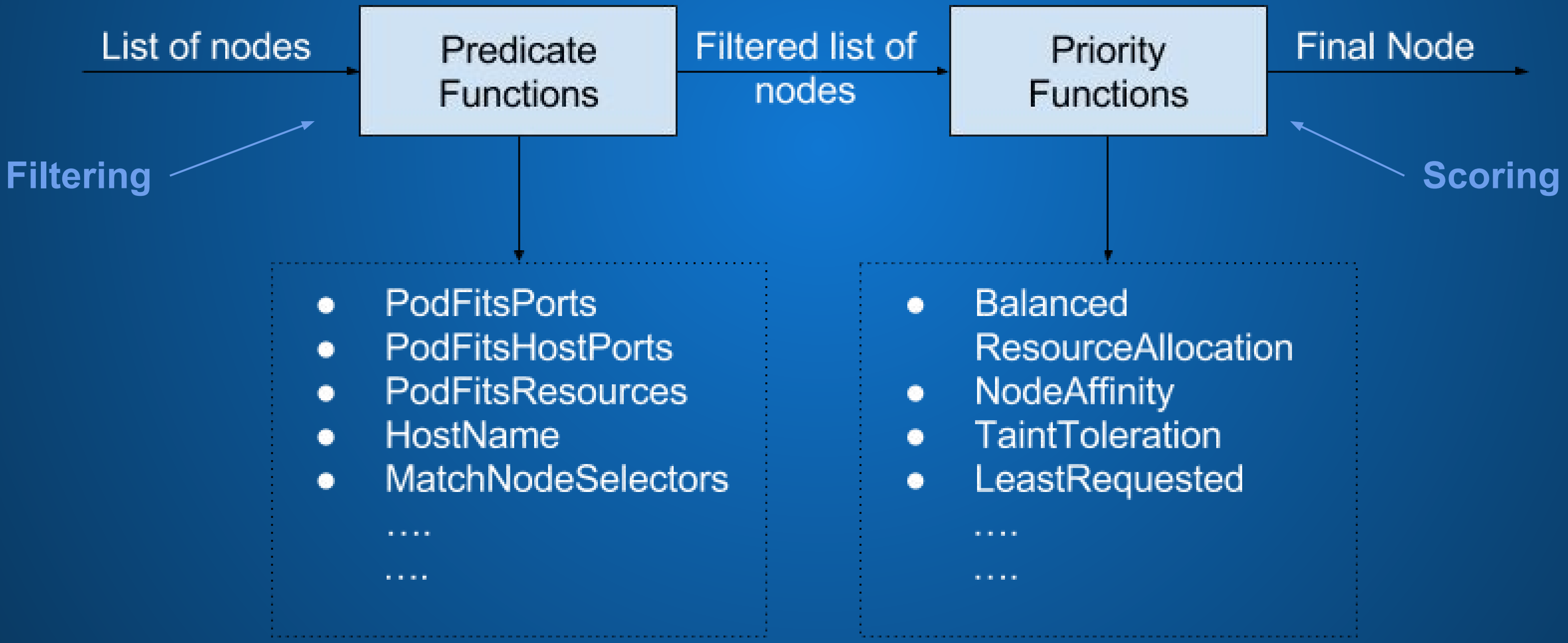
- PodFitsResources: Verify nodes has sufficient memory, cpu and gpu.

- MatchesNodeSelectors: If specified, verify node selectors listed for the pod.

Default Priorities

- Selector Spread: Spread the pods in a service/deployment across nodes.
- ImageLocality: Prefer nodes which already have the container images.
- Least Requested: Nodes with least requested resources (cpu/memory) are preferred
- Node Label: Prefer node if it has a matching label.

In a nutshell...



List of nodes

Predicate Functions

Filtered list of nodes

Priority Functions

Final Node

Filtering

Scoring

- PodFitsPorts
- PodFitsHostPorts
- PodFitsResources
- HostName
- MatchNodeSelectors

....

....

- Balanced ResourceAllocation
- NodeAffinity
- TaintToleration
- LeastRequested

....

....

Drawbacks

- Fixed set of predicates/priorities for all your applications
- No contextual awareness, about application requirements, topology.
- Equally weighted.

How can we do better?

- Prioritize nodes which provide better QoS of an application based on the underlying topology.

- Example: Filter nodes whose available network bandwidth is below a threshold.

- Balancing different requirements by assigning custom weights to different priorities.

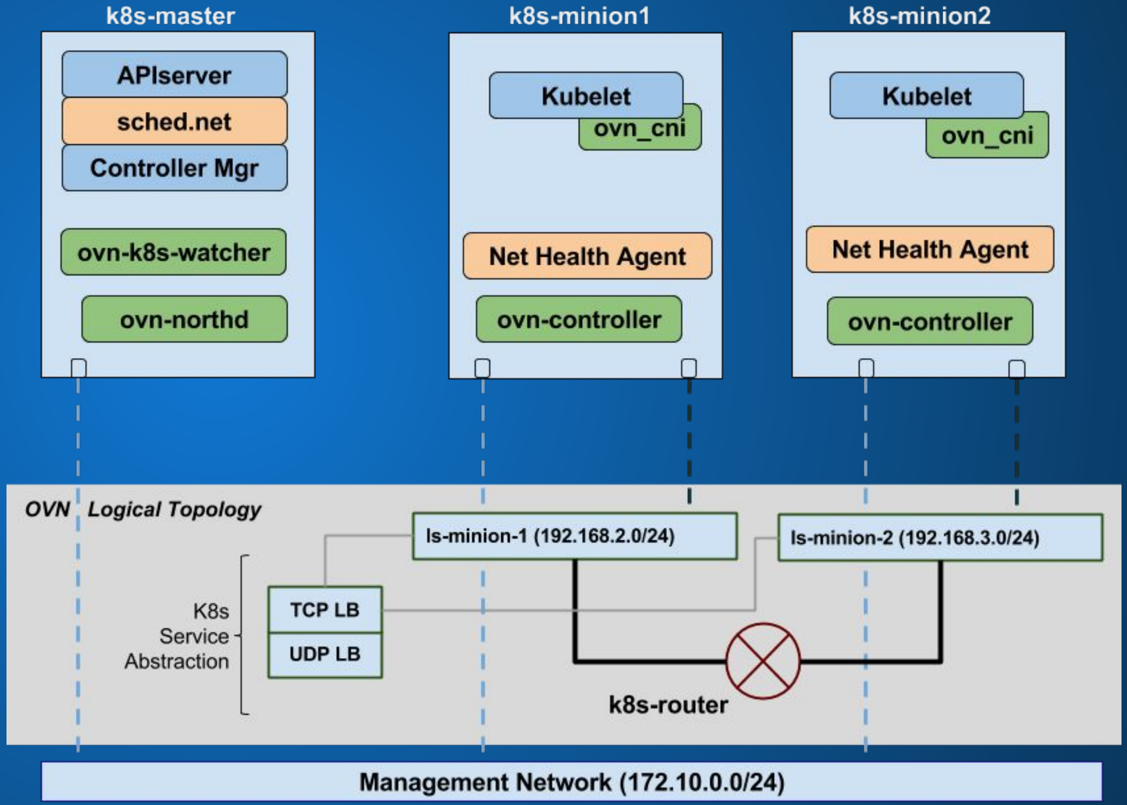
Sched.net

OVN: Open Virtual Network

- Open Source network virtualization solution developed by the Open vSwitch community.
- Allows creation of fundamental networking constructs to build virtual networking topologies:
  - Logical switches
  - Logical routers
  - Stateful ACLs
  - Load-balancers (L4/L7)
- Supported kubernetes networking backend https://github.com/openvswitch/ovn-kubernetes

Experimental testbed - 1/2

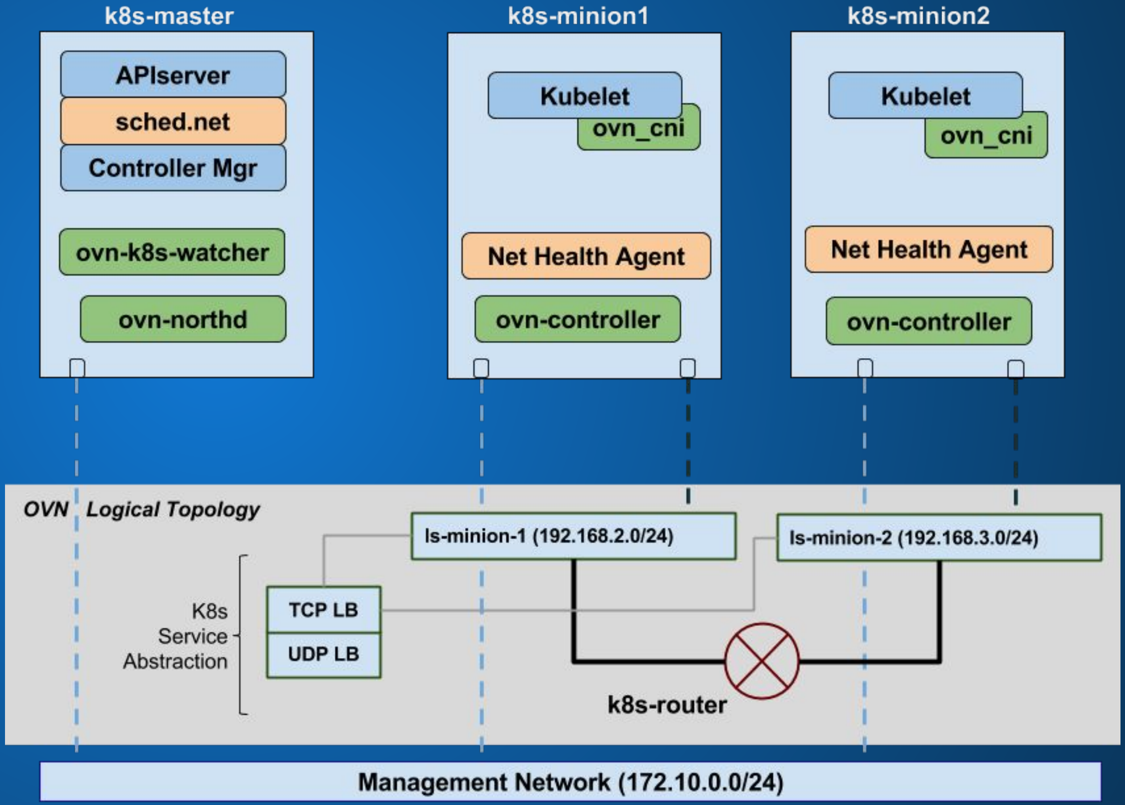
- Distinct logical switch for each K8s node
- All nodes in the cluster connected to the same logical router (DVR)
- Each logical switch associated with TCP & UDP LB for translating cluster IPs
- OVN northbound daemon running on master nodes.
- OVN controllers running on every other node
- Ovn-k8s-watcher monitor events in K8S API Server, configures logical networking for pod
- OVN CNI plugin configures veth pair for Pod network interface, and attaches it to local OVS bridge instance



Experimental testbed - 2/2

- Standard kubernetes setup
  - 1 master, 2 nodes
- No kube-proxy
  - OVN provides the same capabilities

- Sched.net augments kubernetes scheduler with network awareness
- Net Health Agents on every node collects network state data
- Sched.net evaluates scheduling predicates based on this data

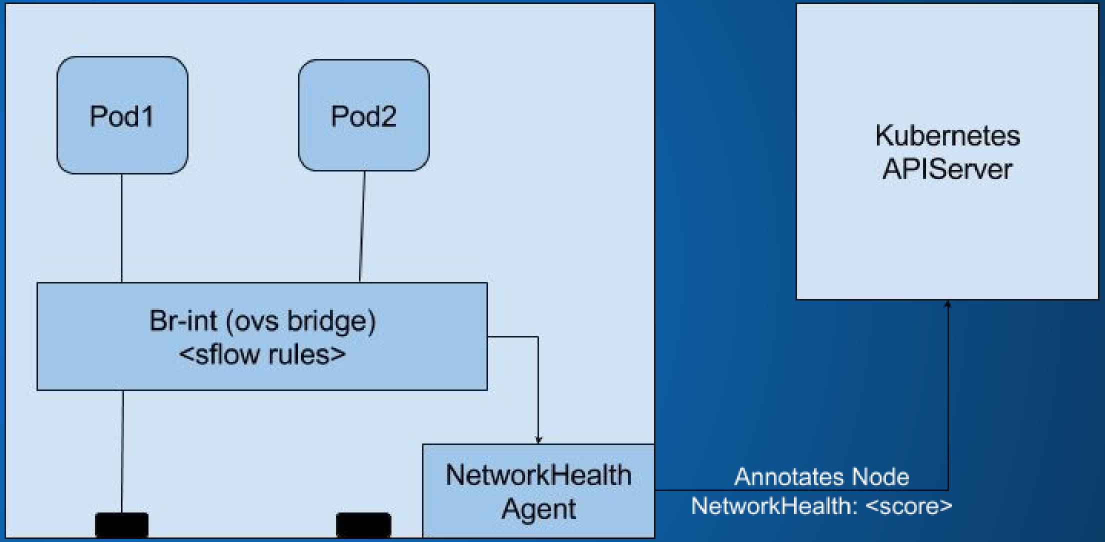


Network Health Agent

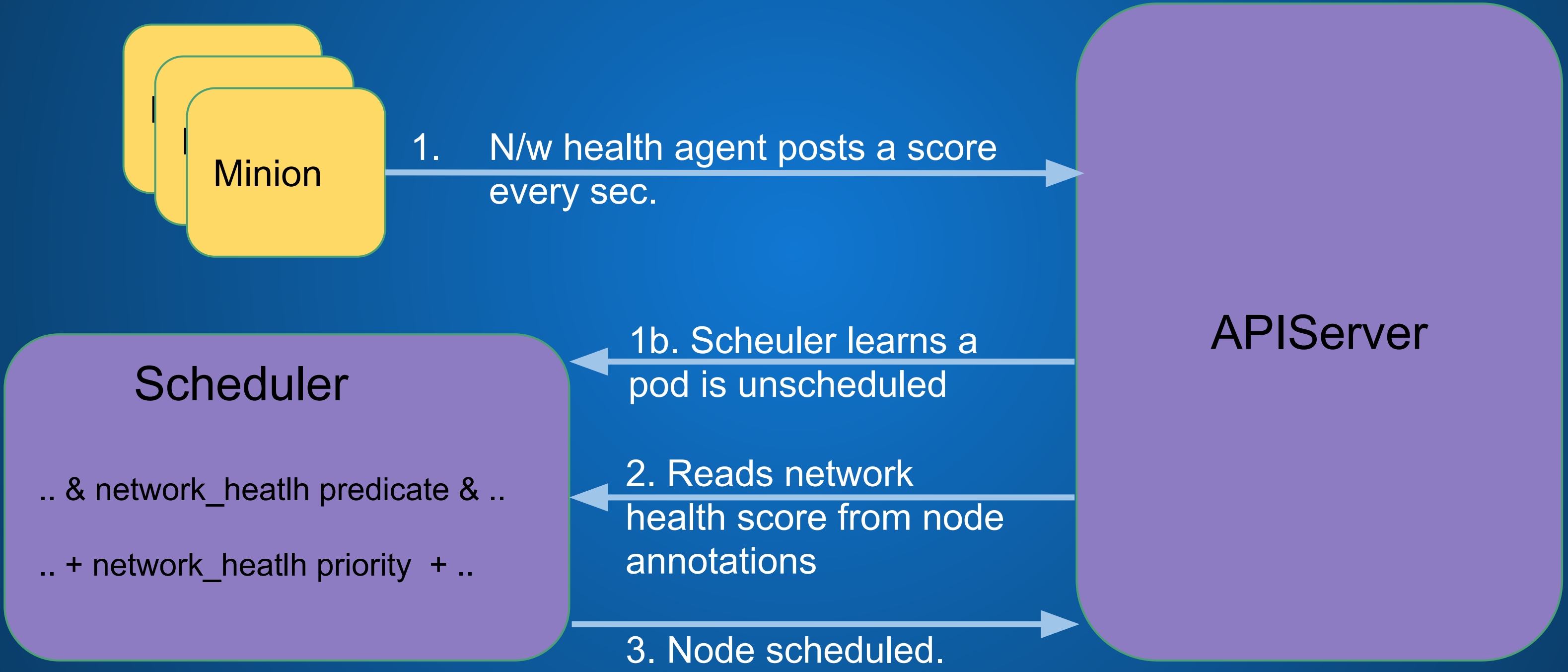
- Collets the network throughput through all the network interfaces on the node via `ifstat`

- /PATCH /api/v1/node

  Network_health: score



How it fits together?



Minion

1. N/w health agent posts a score every sec.

APIServer

1b. Scheuler learns a pod is unscheduled

Scheduler

.. & network_heatlh predicate & ..

.. + network_heatlh priority + ..

2. Reads network health score from node annotations

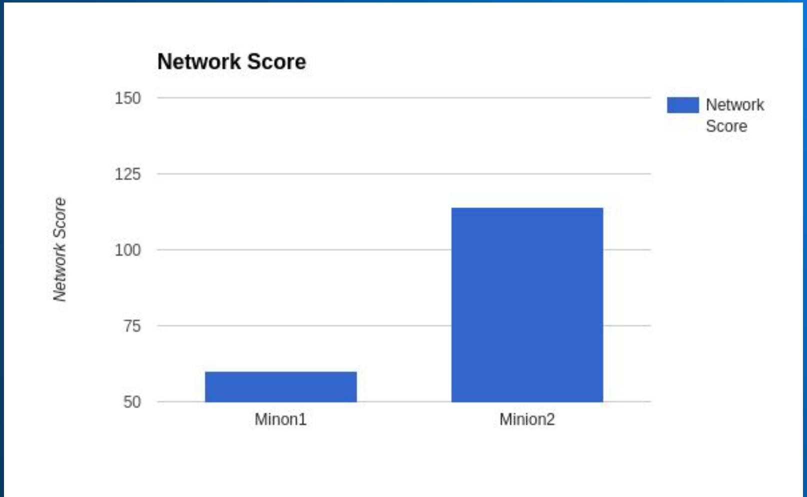
3. Node scheduled.

Demo

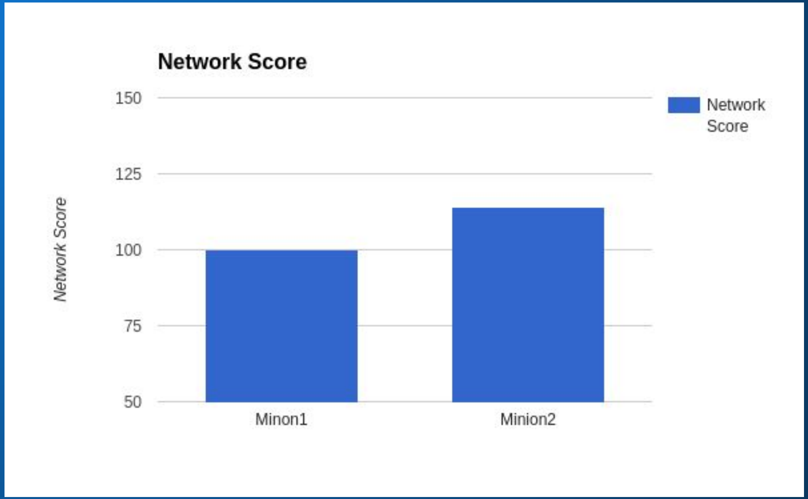
Observations

- 4 pods specs sending 55KB, 30KB, 3KB and 200 B.
- In case of default scheduler, pods are scheduled evenly on both nodes.
- In case of sched.net, pods are scheduled to distribute the network load evenly.

Default Scheduler



Sched.net



Alternative Approach

- Run multiple schedulers in your cluster. Implement custom scheduling algorithm in a separate “scheduler” by implementing the generic_scheduler interface.

PodSpec

```
spec:
  schedulername: my-scheduler
```

- Leverage the scheduler extender interface.

Use Cases

- Leverage insights from from software-defined infrastructure.

- Heterogeneous set of applications.

- Experimentation

Thank you!

Questions?

References

1. https://github.com/kubernetes/community/blob/master/contributors/devel/scheduler.md
2. https://kubernetes.io/docs/admin/multiple-schedulers/