Edge Challenges and Opportunities: Data, Latency, **Resilience**

Professor Richard Mortier

Systems Research Group,
Department of Computer Science & Technology,
University of Cambridge, UK

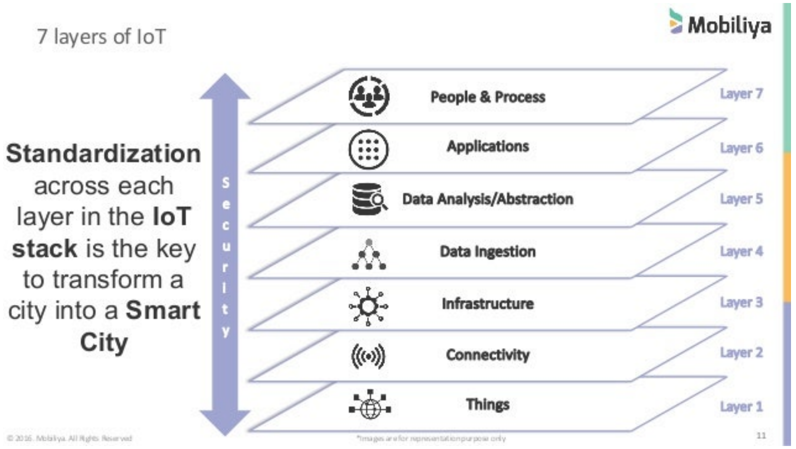
with thanks to: Justas Brazauskas, Matt Danish, **Heidi Howard**, **Andrew Jeffery**, **Chris Jensen**, Roman Kolcun, Ian Lewis, Vadim Safronov, **Alessandro Sassi** (P. de Milano), **Al-Amjad Tawfiq Isstaif**, Rohit Verma

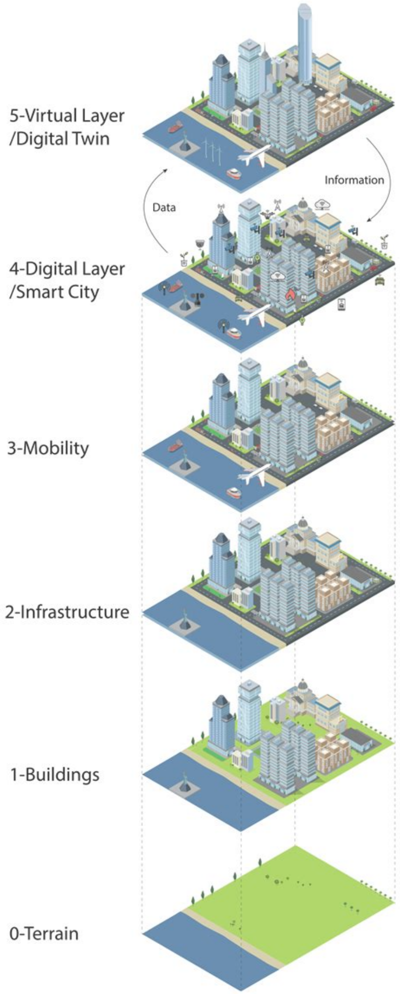
How am I defining the Edge?

- A range of different device types, including
  - Small resource constrained devices – RPis, home gateways, etc
  - Distributed but not so constrained devices – 5G RAN etc
- A range of different network types, including
  - LoRaWAN, Zigbee, SIGFOX
  - 4/5G, Wi-Fi, wired Ethernet
- Instead, define the edge via **common system characteristics**
  - Geographically distributed
  - Relatively limited CPU and memory resources
  - Network connected but potentially constrained and unreliable

Example: Smart Cities

- Connectivity is a fundamental requirement
  - Require low-power, low-latency, low-touch network
  - E.g., LoRaWAN as a lowest-common denominator
    - One LoRaWAN gateway per building vs 10s of Wi-Fi APs
    - Little infrastructure required
- ...but what about latency?
- ...and how to make resilient?
  - Scale and geographical distribution





https://www.researchgate.net/figure/Layers-Required-to-Develop-a-Digital-Twin-Smart-City_fig1_348382801
https://image.slidesharecdn.com/iotsmartcities-170123123747/95/iot-amp-smart-cities-11-638.jpg?cb=1485175097

Edge, Challenges and Opportunities

1) **Data locality**
   - Many applications naturally generate data in a distributed fashion
   - Use this rather than centralise data
     - E.g., Anemone [Mortier et al, 2006], Seaweed [Narayanan et al, 2006] :)
2) **Latency**
   - Latency to the cloud may be lower than you think [Mohan et al, HotNets 2020]
   - But some network types simply can’t support low latency cloud access
3) **Resilience**
   - Infrastructure applications need resilience
   - Must keep working, even if degraded, when nodes, links, services fail

(1) **Data locality** via deployment

- First, **IoT device identification at the edge**
  - Apply a set of pre-trained binary classifiers to identify devices
  - Use the model implied by detection to determine anomalous behaviour
  - Allow for re-training of models using local knowledge

- Second, **Complex event processing at the edge**
  - Synthesis of higher-level events from raw high-frequency sensing data
  - Provides low-latency localised decision making
  - Better fit to the bandwidth constraints of LoRaWAN backhaul

Edge, Challenges and Opportunities

1) **Data locality**
   - Many applications naturally generate data in a distributed fashion
   - Use this rather than centralise data
     - E.g., Anemone [Mortier et al, 2006], Seaweed [Narayanan et al, 2006] :)
2) **Latency**
   - Latency to the cloud may be lower than you think [Mohan et al, HotNets 2020]
   - But some network types simply can’t support low latency cloud access
3) **Resilience**
   - Infrastructure applications need resilience
   - Must keep working, even if degraded, when nodes, links, services fail

(2) Latency, via localised compute

- First, **A smart camera performing object recognition**
  - Turns high bandwidth video stream into low bandwidth object counts
  - *“3 cars, 2 people and a bus”* or *“at time T, a person entered the building”*

- Second, **A rearchitecting of LoRaWAN for low latency**
  - Avoid backhauling all data to a central location before acting
  - Remove IP from the stack to improve performance

Edge, Challenges and Opportunities

1) **Data locality**
   - Many applications naturally generate data in a distributed fashion
   - Use this rather than centralise data
     - E.g., Anemone [Mortier et al, 2006], Seaweed [Narayanan et al, 2006] :)
2) **Latency**
   - Latency to the cloud may be lower than you think [Mohan et al, HotNets 2020]
   - But some network types simply can’t support low latency cloud access
3) **Resilience**
   - Infrastructure applications need resilience
   - Must keep working, even if degraded, when nodes, links, services fail

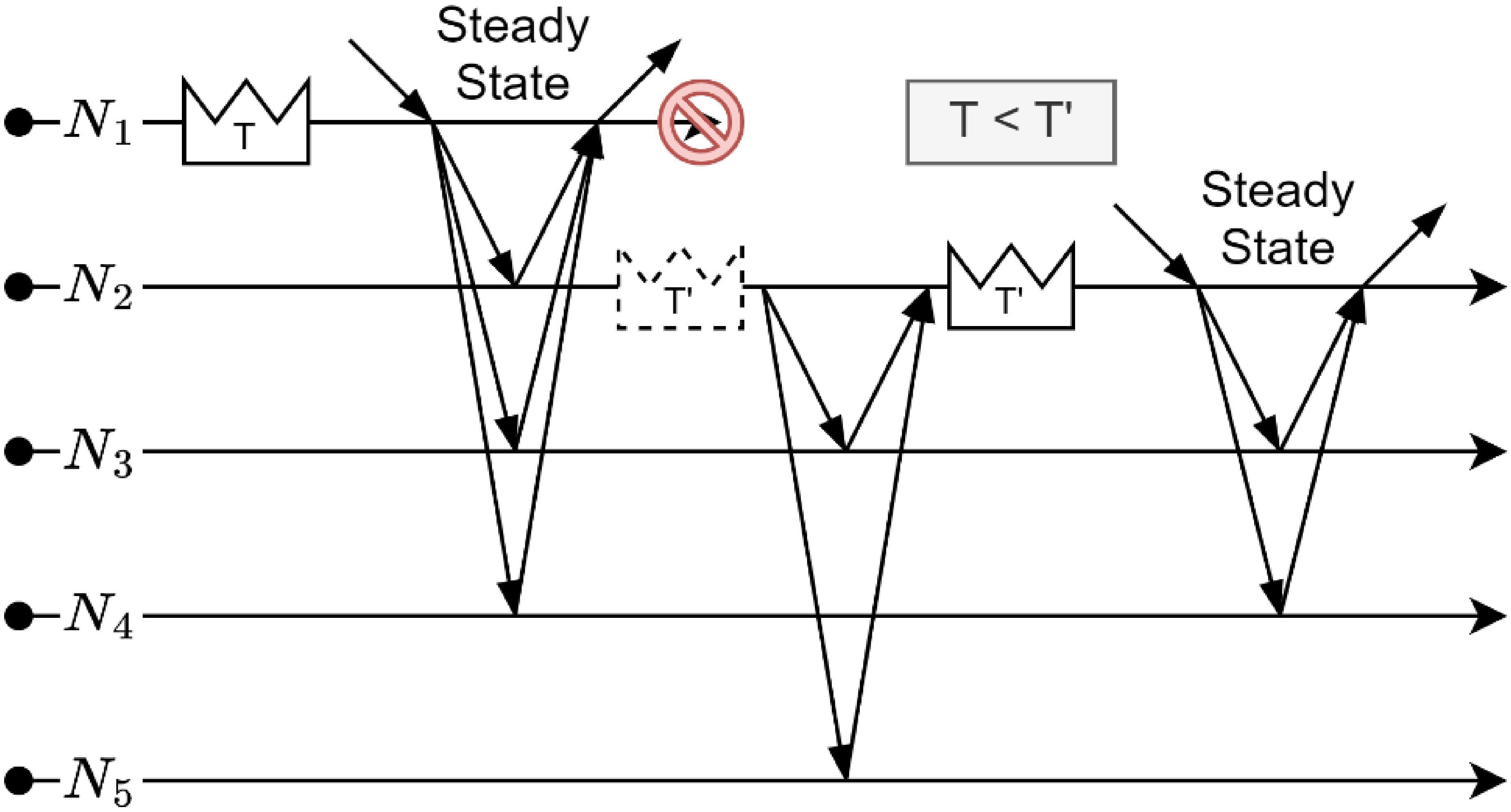
(3) **Resilience**, via distribution

- First, **Understand better how orchestration behaves at edge scale**
  - Most rely on consensus systems rarely deployed beyond 1/3/5 node clusters!
- Then, **Extend orchestrator to improve resilience**
  - Get the benefits of Paxos in the more popular Raft
- Finally, **Revisit assumptions to better target the edge**
  - Radical changes require careful modelling to ensure correct behaviour

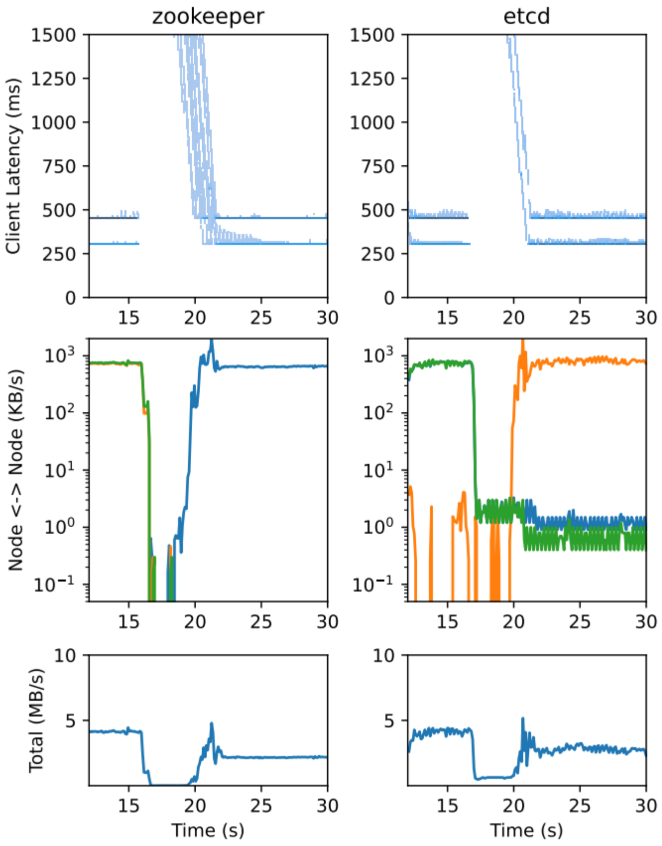
- Ultimately,

eventual consistency works and scales better than strict consistency

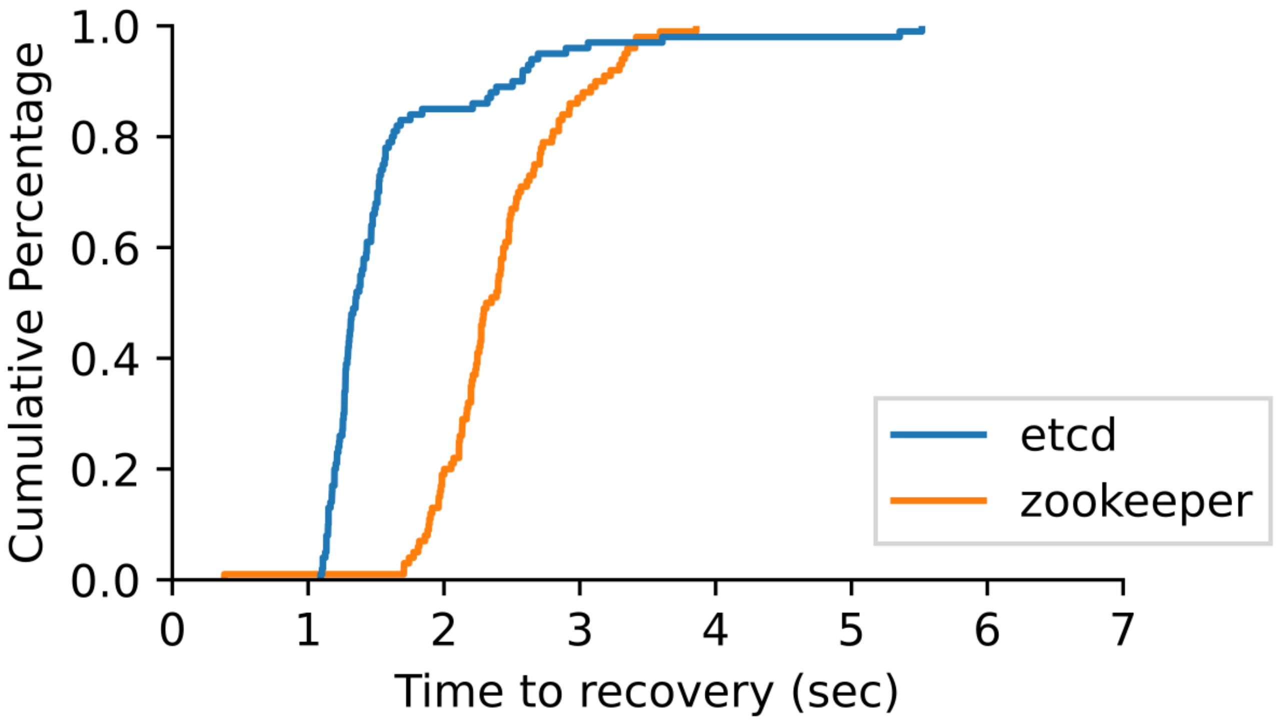
Orchestration relies on consensus



Leader is a single point of failure

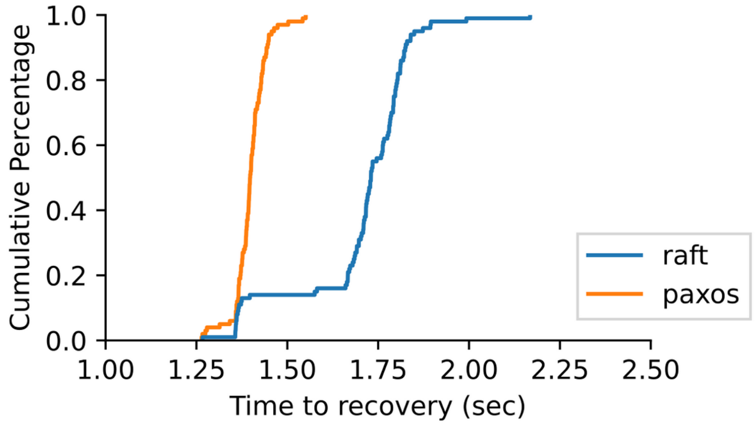


not great in a system designed for resilience!

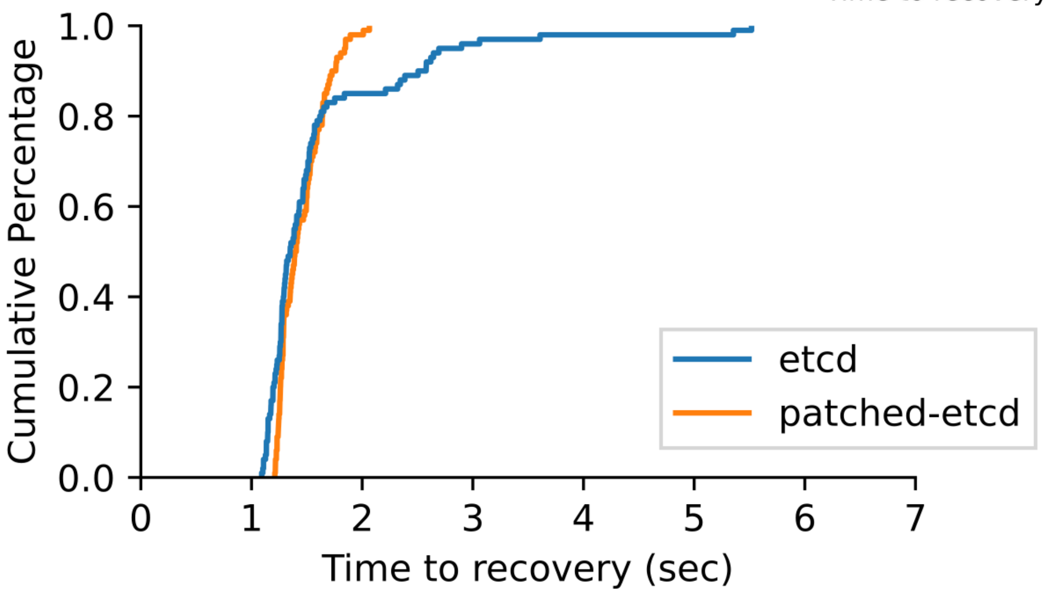


Raft popular, Paxos better?

- Raft leader election can *duel* => long tail latencies
    - Majority voting vs static term ordering
- Fix by patching Raft to randomise low bits of term

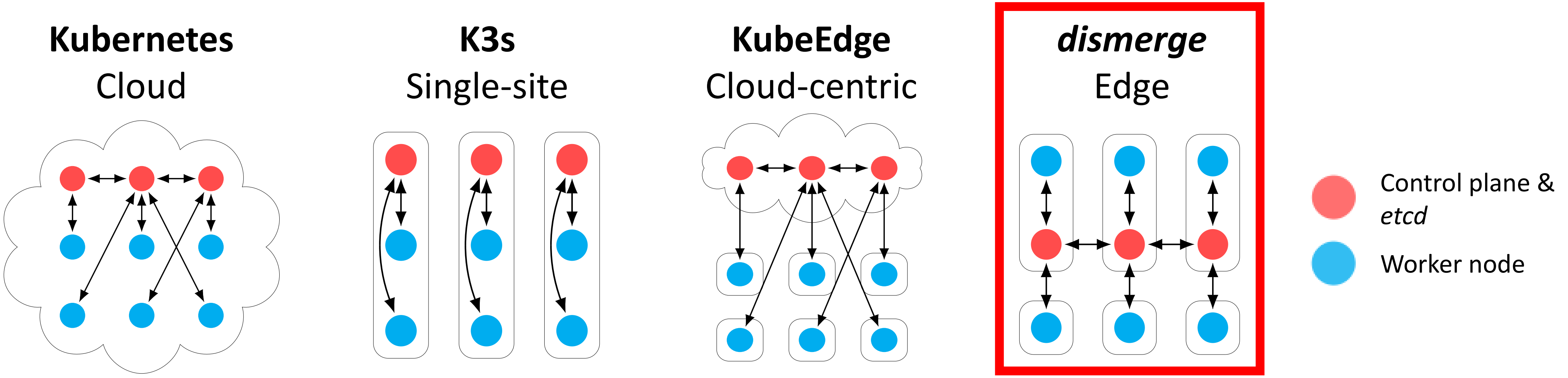



```
diff --git a/raft.go b/raft.go
index d104829..e8eb5bd 100644
--- a/raft.go
+++ b/raft.go
@@ -840,0 +841,8
+func (r *raft) nextTerm() uint64 {
+  // Term = [epoch:48; rand:16]
+  var cepoch uint64 = (r.Term & 0xffff_ffff_ffff_0000) >> 16
+  var tepoch uint64 = (cepoch + 1) << 16
+  var trdm uint64   = uint64(globalRand.Intn(65536)) & 0xffff
+  return tepoch | trdm
+}
+
@@ -847 +855 @@ func (r *raft) becomeCandidate() {
-       r.reset(r.Term + 1)
+       r.reset(r.nextTerm())
@@ -946 +954 @@ func (r *raft) campaign(t CampaignType) {
-               term = r.Term + 1
+               term = r.nextTerm()
```



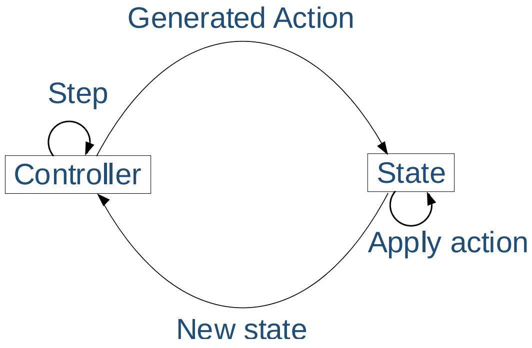
Distributing orchestration

- How to use edge resources in a cluster while maintaining resilience?
  - Avoiding both isolation of resources and enlarging the failure's blast radius



- How to ensure correct behaviour of Kubernetes upon such a radical change?

Modelling orchestration



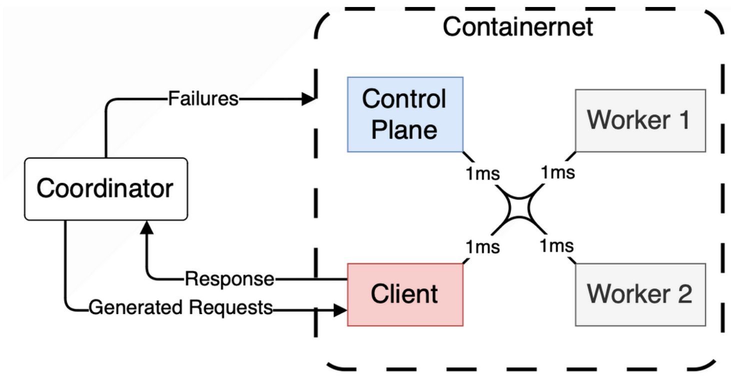

```
{
  "pods": {
    "llama2": {
      "image": "hf/llama:2"
      "node": "node-0",
    }
  },
  "nodes": {
    "node-0": ["llama2"]
  },
}
```

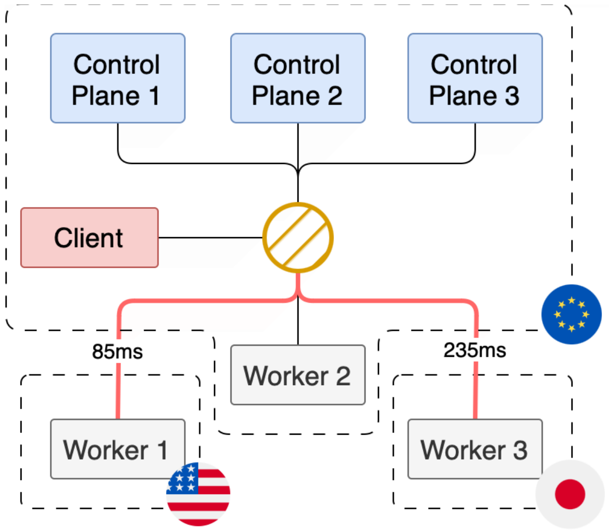

- `CreatePod("llama2")`
- `SchedulePod("llama2", "node-0")`
- `RunPod("node-0", "llama2")`

Controller	Covered lines	Total lines	Percentage
Scheduler	52	78	66.67
Job	339	760	44.61
ReplicaSet	151	204	74.02
Deployment	579	909	63.7
StatefulSet	470	687	68.41

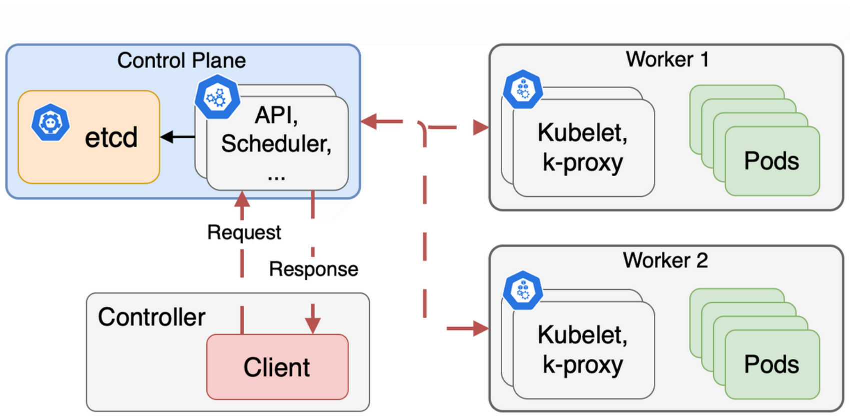
- Model controller behaviour as stepping forward from starting state, generating and applying actions
- Extract *properties* from Kubernetes integration tests, documentation, and “well-known” behaviour
- Reimplement relevant controllers in Rust and apply the *stateright* model checking library to explore whether properties hold
  - Simulation-based exploration of different configurations of controllers

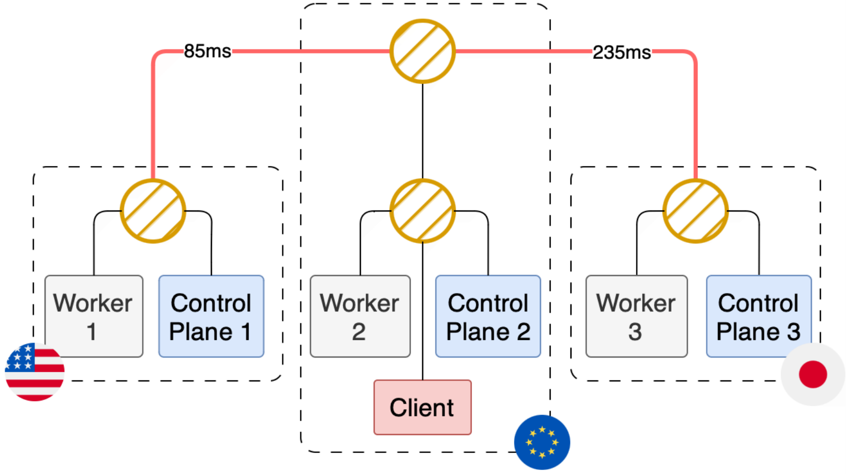
Simulating orchestration



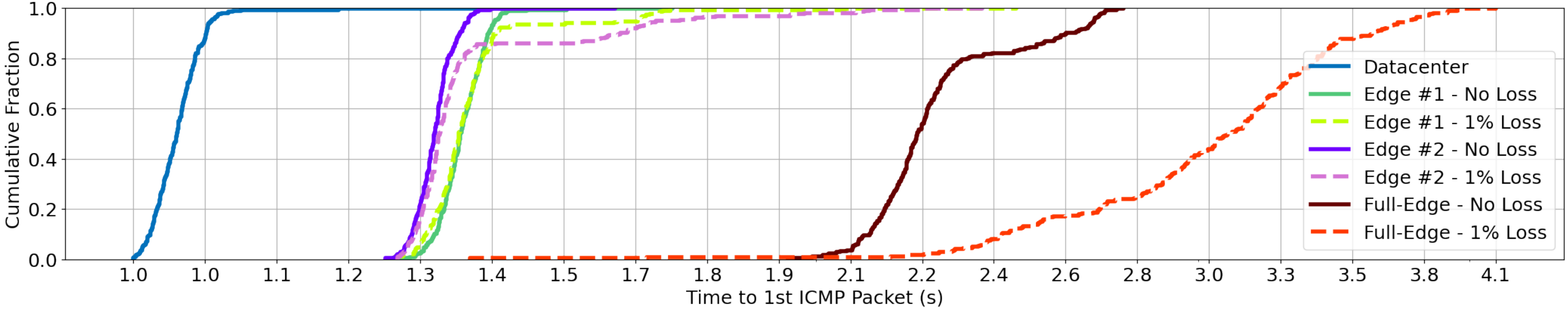


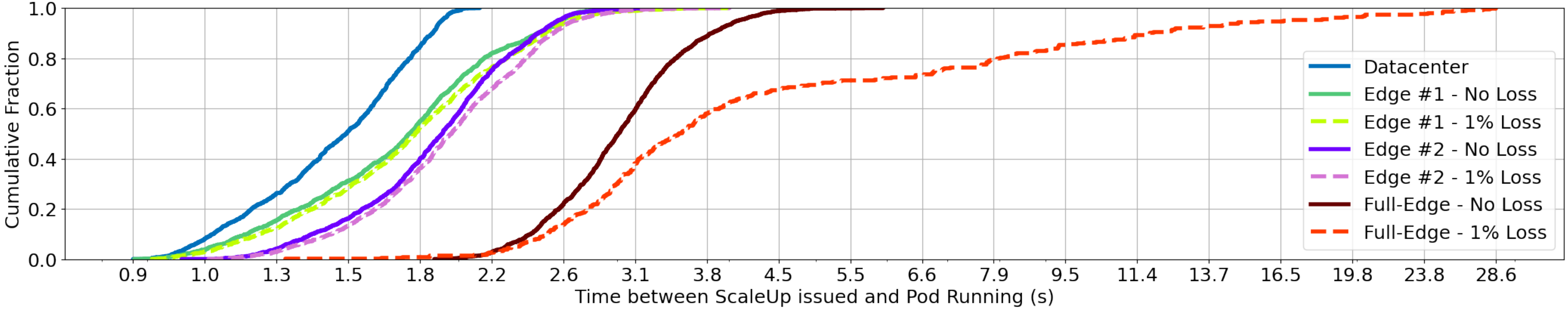
- Use *kind* and *containernet* to emulate Kubernetes deployment over a controllable network
- External *coordinator* provokes a *client* to issue requests and injects failures to the deployment
- Examine multiple configurations distributing workers and control plane





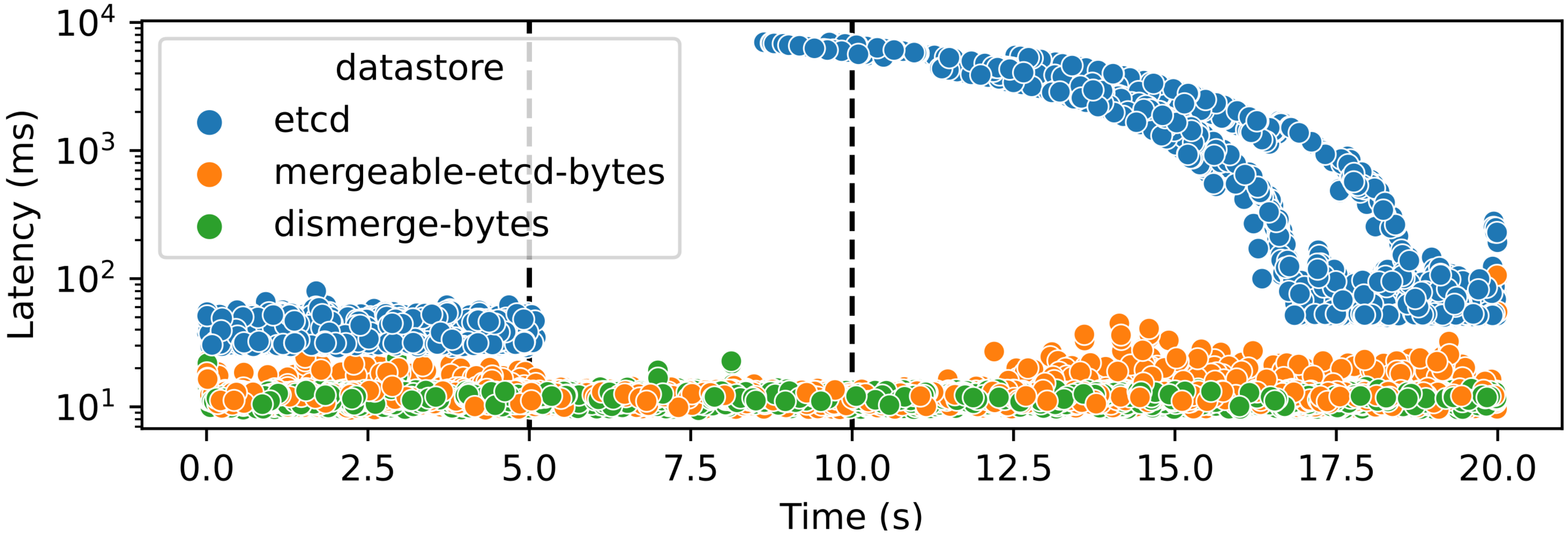
Simulating orchestration





Distributing orchestration

- Replace opaque strictly-consistent key-value store with a Conflict-free Replicated Data Type such as an *Automerge* JSON document
- Make every node a leader, resolve discrepancies in JSON on merge



3 nodes, leader partitioned between t=5 and t=10, 10ms link delay, successful requests, 10,000 rps

Questions!

- Smart Camera aka “DeepDish”
  - *DeepDish: multi-object tracking with an off-the-shelf Raspberry Pi*, Danish et al, ACM EDGESYS 2020
  - *DeepDish on a diet: low-latency, energy-efficient object-detection and tracking at the edge*, Danish et al, ACM EDGESYS 2022
  - *Anonymising Video Data Collection at the Edge Using DeepDish*, Pan et al, IEEE HPSR 2023
- Consensus & Orchestration
  - *Paxos vs Raft: Have we reached consensus on distributed consensus?*, Howard et al, ACM PaPoC 2020
  - *Rearchitecting Kubernetes for the Edge*, Jeffery et al, ACM EDGESYS 2021
  - *Examining Raft’s behaviour during partial network failures*, Jensen et al, ACM HAOC 2021
  - *AMC: Towards Trustworthy and Explorable CRDT Applications with the Automerge Model Checker*, Jeffery et al, ACM PAPOC 2023
- Networking
  - *Do we want the New Old Internet?: Towards Seamless and Protocol-Independent IoT Application Interoperability*, Safronov et al, ACM HOTNETS 2021
  - *Revisiting IoT Device Identification*, Kolcun et al, IFIP TMA 2021
- Smart Cities
  - *RACER: Real-Time Automated Complex Event Recognition in Smart Environments*, Verma et al, ACM SIGSPATIAL 2021
  - *Real-time data visualisation on the adaptive city platform*, Brazauskas et al, ACM BuildSys 2021
  - *CDBB West Cambridge Digital Twin: Lessons Learned*, Brazauskas et al, arXiv:2209.15290 2022