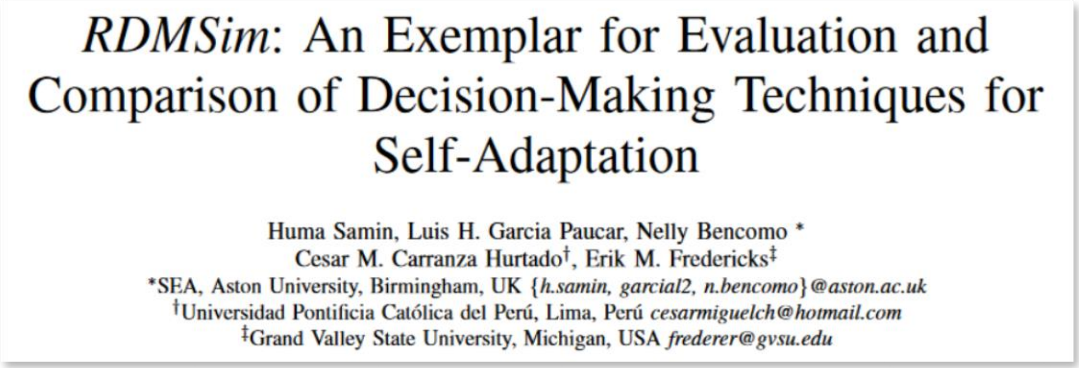
Dr.-Ing. Sebastian Götz

A Time-aware Remote Data Mirroring Simulator

September 2023

Motivation

- Existing SEAMS Exemplar
- *Application*: Remote Data Mirroring
- *Problem*: No explicit object net of mirrors and links
  - Only calculated metrics for each timestep
- *Consequence*: No way to investigate the net
- *Goal*: Enable detailed investigation and self-adaptation
- *Solution*: Explicit object net



RDMSim: An Exemplar for Evaluation and Comparison of Decision-Making Techniques for Self-Adaptation

Huma Samin, Luis H. Garcia Paucar, Nelly Bencomo [*]
Cesar M. Carranza Hurtado[†], Erik M. Fredericks[‡]
[*]SEA, Aston University, Birmingham, UK {*h.samin, garcial2, n.bencomo*}*@aston.ac.uk*
[†]Universidad Pontificia Católica del Perú, Lima, Perú *cesarmiguelch@hotmail.com*
[‡]Grand Valley State University, Michigan, USA *frederer@gvsu.edu*

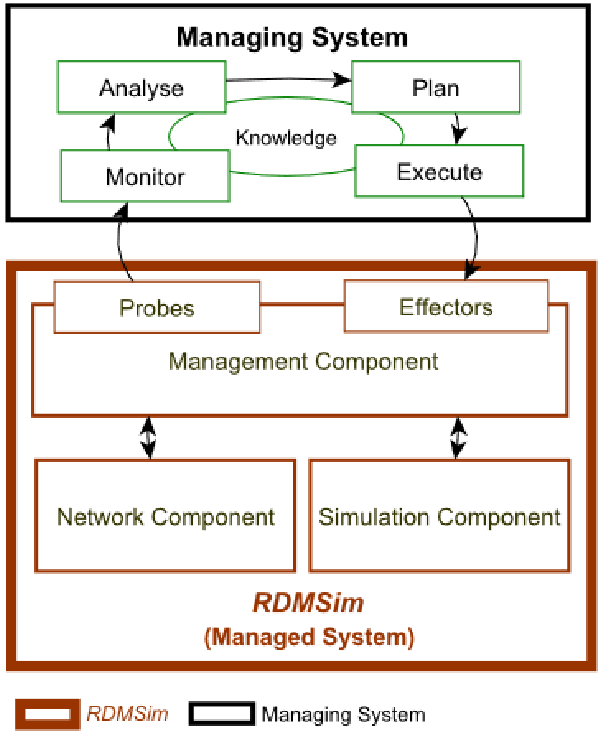


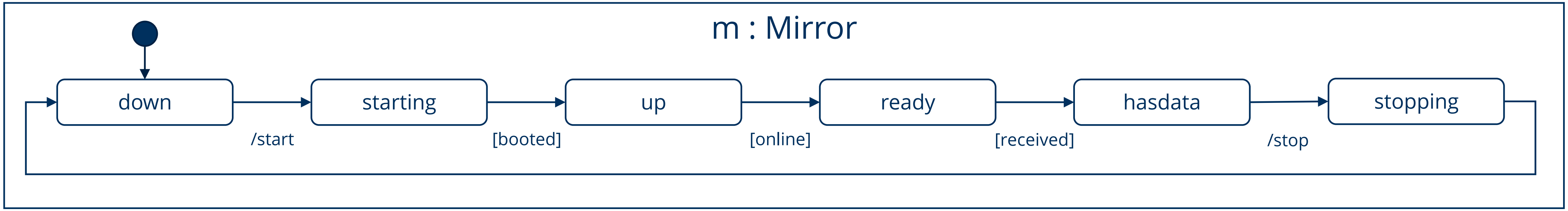
Fig. 1. RDMSim Architecture

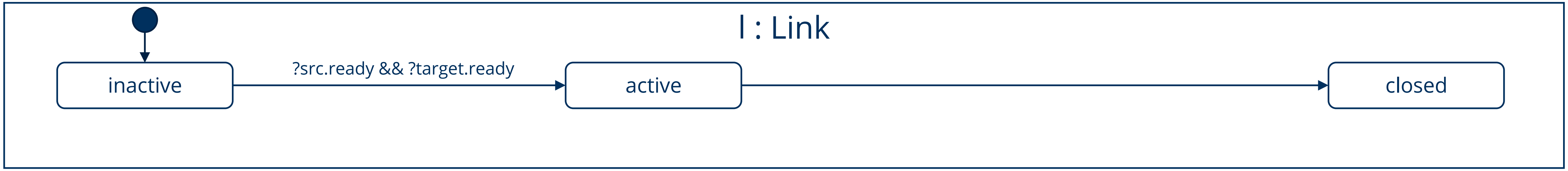
Principle Idea

- Create a network of mirrors
- Connect the mirrors according to a strategy
- Mirrors and Links have states
- State changes are timed
  - Random individual times within given bounds

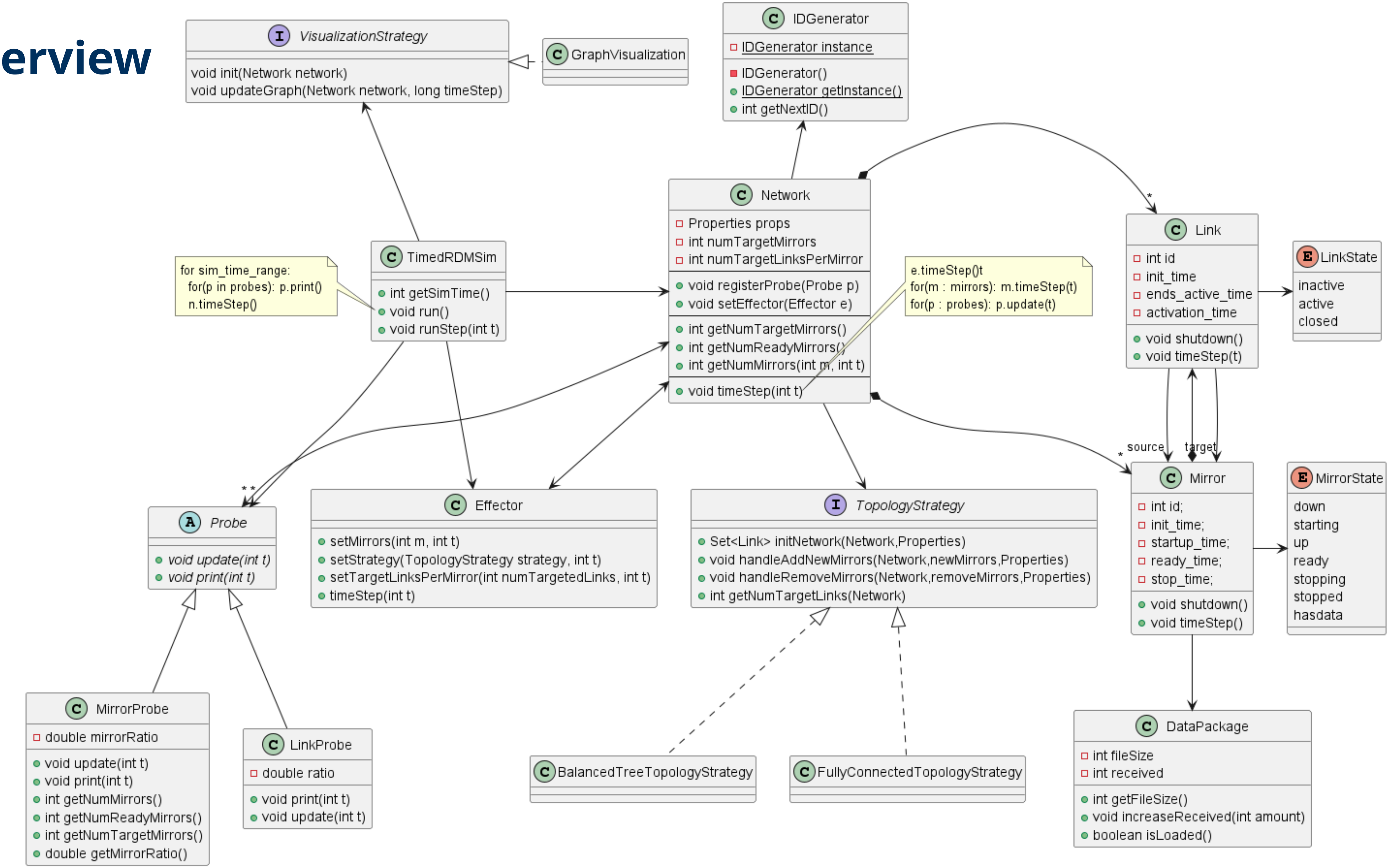
- At runtime a data package is distributed among all mirrors
- At runtime we can change the number of mirrors, the topology and the number of links
- At runtime we can introduce failures
- We can observe t

Link and Mirror States





Framework Overview



I VisualizationStrategy
- void init(Network network)
- void updateGraph(Network network, long timeStep)

C GraphVisualization

C IDGenerator
- IDGenerator instance
- IDGenerator()
- IDGenerator getInstance()
- int getNextID()

C TimedRDMSim
- int getSimTime()
- void run()
- void runStep(int t)

```
for sim_time_range:
  for(p in probes): p.print()
  n.timeStep()
```

C Network
- Properties props
- int numTargetMirrors
- int numTargetLinksPerMirror
- void registerProbe(Probe p)
- void setEffector(Effector e)
- int getNumTargetMirrors()
- int getNumReadyMirrors()
- int getNumMirrors(int m, int t)
- void timeStep(int t)

```
e.timeStep()t
for(m : mirrors): m.timeStep(t)
for(p : probes): p.update(t)
```

C Link
- int id
- init_time
- ends_active_time
- activation_time
- void shutdown()
- void timeStep(t)

E LinkState
- inactive
- active
- closed

A Probe
- *void update(int t)*
- *void print(int t)*

C Effector
- setMirrors(int m, int t)
- setStrategy(TopologyStrategy strategy, int t)
- setTargetLinksPerMirror(int numTargetedLinks, int t)
- timeStep(int t)

I TopologyStrategy
- Set<Link> initNetwork(Network,Properties)
- void handleAddNewMirrors(Network,newMirrors,Properties)
- void handleRemoveMirrors(Network,removeMirrors,Properties)
- int getNumTargetLinks(Network)

C Mirror
- int id;
- init_time;
- startup_time;
- ready_time;
- stop_time;
- void shutdown()
- void timeStep()

E MirrorState
- down
- starting
- up
- ready
- stopping
- stopped
- hasdata

C MirrorProbe
- double mirrorRatio
- void update(int t)
- void print(int t)
- int getNumMirrors()
- int getNumReadyMirrors()
- int getNumTargetMirrors()
- double getMirrorRatio()

C LinkProbe
- double ratio
- void print(int t)
- void update(int t)

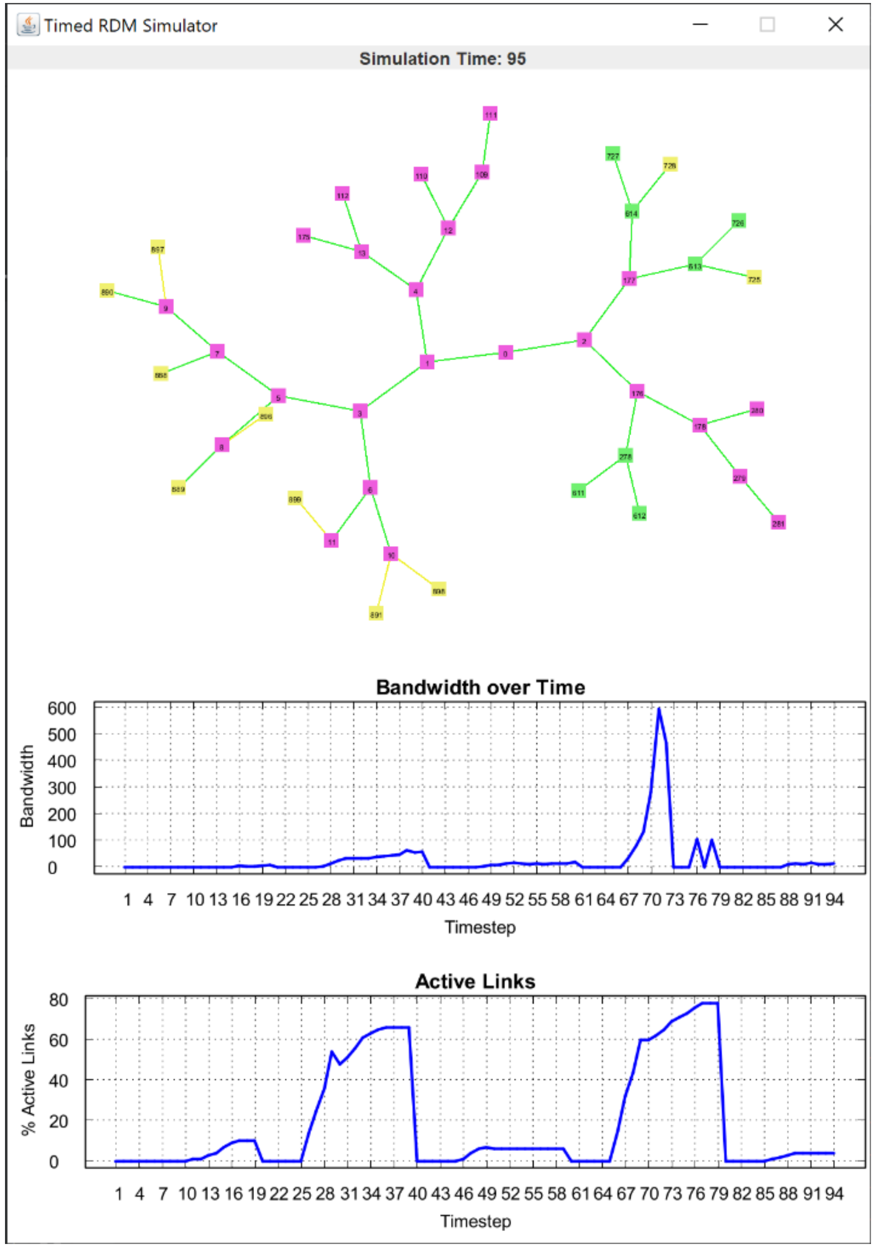
C BalancedTreeTopologyStrategy

C FullyConnectedTopologyStrategy

C DataPackage
- int fileSize
- int received
- int getFileSize()
- void increaseReceived(int amount)
- boolean isLoaded()

Visualization of Network

- With GraphStream 2.0
- State of mirrors and links is color coded
- Yellow: inactive/starting
- Green: active/ready
- Red: closed/stopping
- Purple: hasdata



Timed RDM Simulator

Simulation Time: 95

Simulation Configuration (sim.conf)

```
debug=true
sim_time=200
num_mirrors=10
num_links_per_mirror=2
startup_time_min=5
startup_time_max=10
ready_time_min=2
ready_time_max=20
stop_time_min=2
stop_time_max=5
link_activation_time_min=5
link_activation_time_max=10
fileSize=80
min_bandwidth=2
max_bandwidth=8
```

How to describe scenarios?

- Write a Simulation Runner (main)

```java
public static void main(String args[])
{
  TimedRDMSim sim = new TimedRDMSim(„resources/sim.conf“);
  sim.initialize(new BalancedTreeTopologyStrategy());
  Effector effector = sim.getEffector();
  //change mirrors to 10 at timestep 40
  effector.setMirrors(10, 40);
  //change to fully connected topology at timestep 70
  effector.setTopology(new FullyConnectedTopologyStrategy(), 70);
  sim.run();
}
```

Self-adaptation

- The system can be controlled via its effector and observed via probes
- Two important metrics
  - Number of active links → reliability
  - Bandwidth used by the overall network → cost
- A fully connected topology leads to the best reliability, but also to the highest cost
- A balanced tree toplogy leads to less reliability, but also less cost
- The number of links per mirror allows to in- or decrease reliability and cost

What to do with it?

- Test self-adaptive controllers with it
  - Could become an easy teaching example for SESAC
- Extend data packages to models and investigate consistency strategies
  - Different strategies to propagate the „dirty“ flag (e.g., push vs. pull)
  - Different strategies to update the models (delta, full, …)
    - You can observe the used bandwidth and timesteps required until consistency is reached!
  - Investigate robustness in presence of failures
- Investigate Fidelity of Models@run.time
  - How to capture the time required until runtime model and managed system are consistent

Thanks

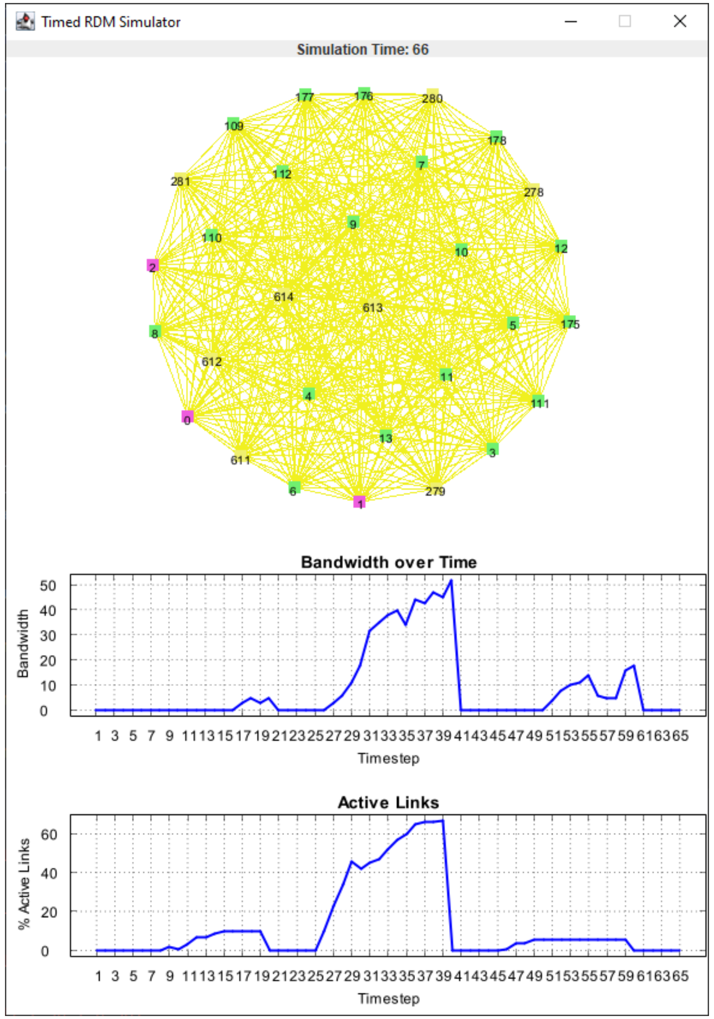
Interested?

Visit

https://github.com/sebastiangoetz/TimedRDMSimulator

Join me in writing the SEAMS'24 exemplar paper.

Deadline: 15.12.2023


Timed RDM Simulator
Simulation Time: 66
177
176
280
109
178
7
112
281
278
9
110
10
12
2
614
613
175
5
8
612
11
4
111
0
13
3
611
6
1
279
Bandwidth over Time
Bandwidth
50
40
30
20
10
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65
Timestep
Active Links
% Active Links
60
40
20
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65
Timestep