OAK RIDGE National Laboratory

Shortest Path and Neighborhood Subgraph Extraction on a Spiking Memristive Neuromorphic Implementation

Catherine D. Schuman, Kathleen Hamilton, Tiffany Mintz, Md Musabbir Adnan, Bon Woong Ku, Sung-Kyu Lim and Garrett S. Rose

Oak Ridge National Laboratory, University of Tennessee and Georgia Institute of Technology

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

TENNLab Neuromorphic Research

THE UNIVERSITY OF TENNESSEE KNOXVILLE

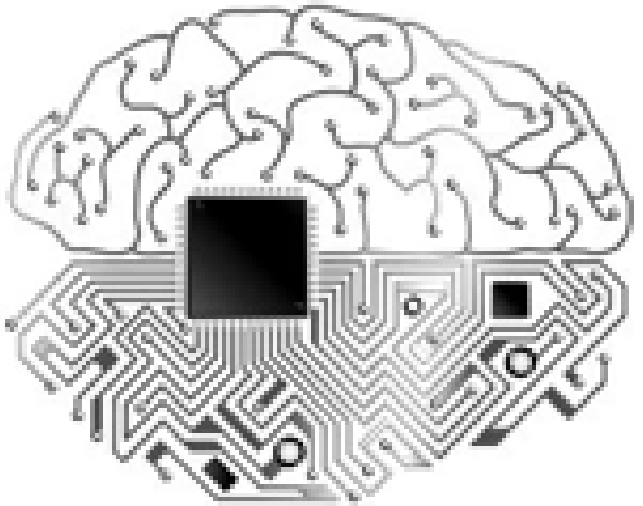
OAK RIDGE National Laboratory

neuromorphic.eecs.utk.edu

- **Applications**
- **Algorithms**
- **System Software and Communications**
- **System Architecture/Organization**
- **Microarchitecture**
- **Devices**
- **Materials**

What are Spiking Neuromorphic Systems?

- Non von-Neumann computers inspired by biological brains
- Can perform certain workloads with significantly less power than traditional architectures
- Resurgence in neuromorphic systems in recent years due to the rise of machine learning and deep learning applications



Source: https://www.nextplatform.com/2016/02/09/the-second-coming-of-neuromorphic-computing/

Properties of Spiking Neuromorphic Systems

- Massively parallel computation
- Collocated processing and memory
- Simple processing elements that perform specific computations
- Simple communication between elements
- Event driven computation
- Stochastically firing neurons for noise
- Inherently scalable architectures

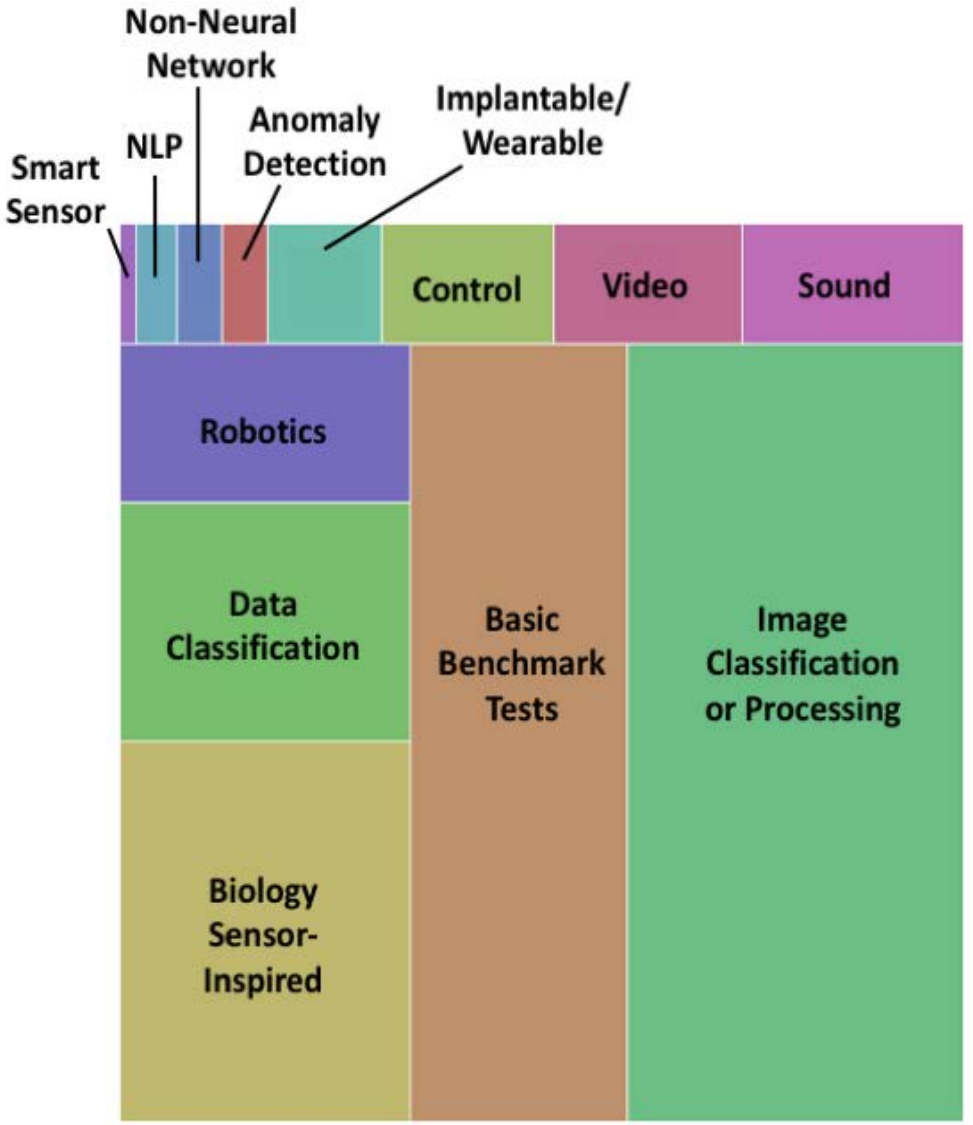
Key research question:
Can we use these properties for non-neural network and non-machine learning applications?

Why do we care about non-neural network applications?

?

How are spiking neuromorphic systems used?

- Through January 2017, less than 1 percent of the papers in neuromorphic computing that implemented an application did so with a non-neural network application



Source: Schuman, Catherine D., et al. "A survey of neuromorphic computing and neural networks in hardware." *arXiv preprint arXiv:1705.06963* (2017).

Example Non-Neural Network Applications

- Graph algorithms
  - See Abdullahi Ali's poster on network flow
- Scientific simulations
- Constraint satisfaction and optimization problems
  - Several talks earlier this week (e.g., Sudoku and graph coloring)
- Solving partial differential equations through neural algorithms and Markov random walks
  - See Leah Reeder's poster and Brad Aimone's talk yesterday
- Composite algorithms through utility and numerical kernels

Memristive Spiking Neuromorphic System

- Leaky integrate-and-fire neurons with programmable thresholds and refractory periods
- Synapses with programmable delays and weights, and some form of STDP

	Accumulation	Fire	Learning	Idle
Neuron	9.81 pJ	12.5 pJ	-	7.2 pJ
Synapse	1.45 pJ	-	2.58 pJ	0.07 pJ

Source:

- M. M. Adnan, S. Sayyaparaju et al., “A twin memristor synapse for spike timing dependent learning in neuromorphic systems,” in Proceedings of the 31st IEEE International System-On-Chip Conference (SOCC). IEEE, 2018. To appear.
- G. Chakma, M. M. Adnan et al., “Memristive mixed-signal neuromorphic systems: Energy-efficient learning at the circuit-level,” IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 8, no. 1, pp. 125–136, 2018.
- C. D. Schuman, R. Pooser et al., “Simulating and estimating the behavior of a neuromorphic co-processor,” in Proceedings of the Second International Workshop on Post Moores Era Supercomputing. ACM, 2017, pp. 8–14.

How do you map a problem onto neuromorphic systems?

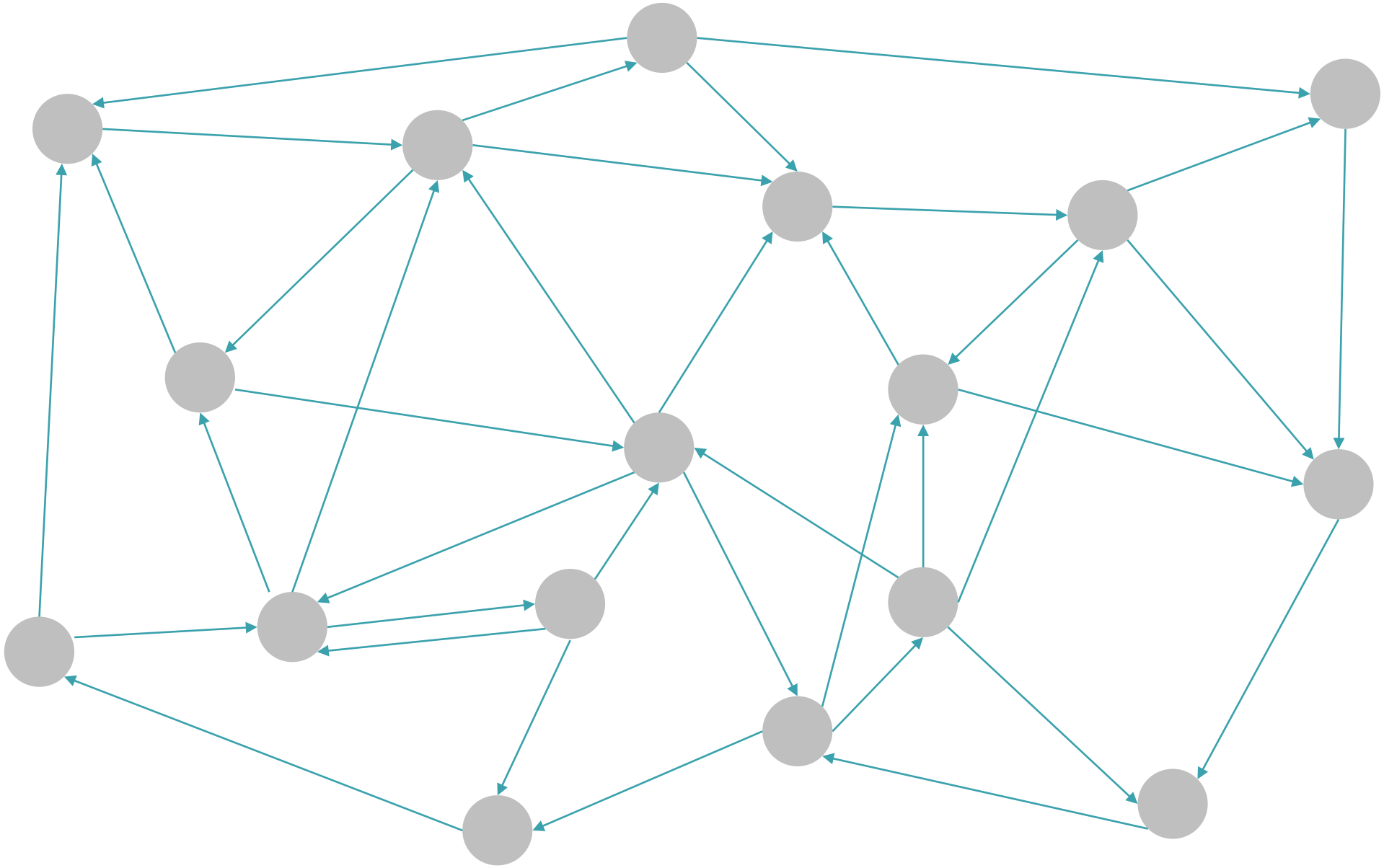
- Need to convert your problem into using neurons, synapses, and spikes
- Need to define:
  - How to set up your network (network topology, parameters)
  - Where/when spikes should be applied
  - How long to run your neuromorphic systems
  - Where/when to read and write network properties
- We'll walk through two graph problems:
  - Single source shortest path
  - Neighborhood subgraph extraction

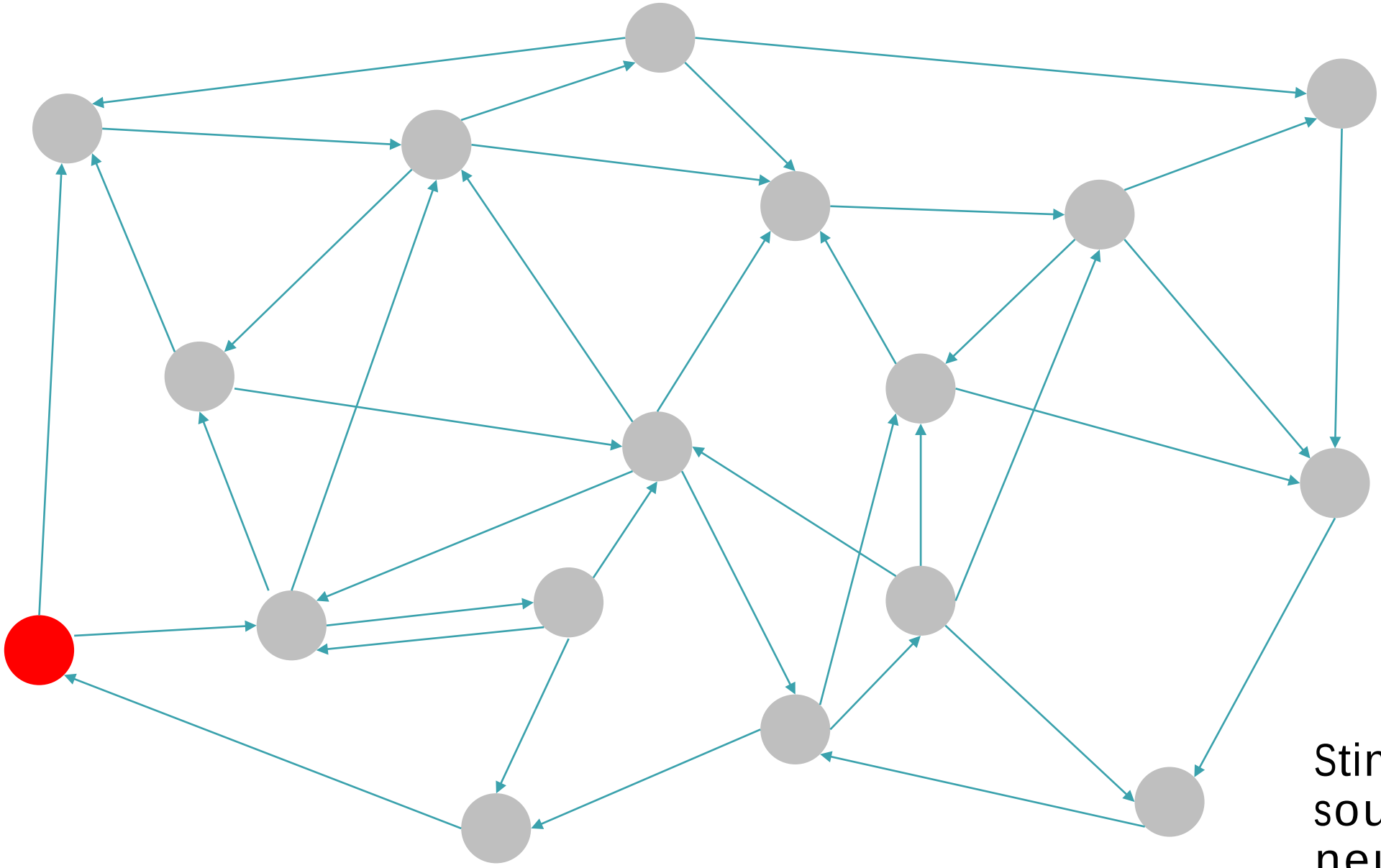
Single Source Shortest Path Embedding

- Graph G to Network N:
  - Vertex in G → Neuron in network N with 0 threshold and a long enough refractory period to make sure it fires only once
  - Edge in G → Synapse in network N with weight of 1 and delay proportional to the length of the edge in G

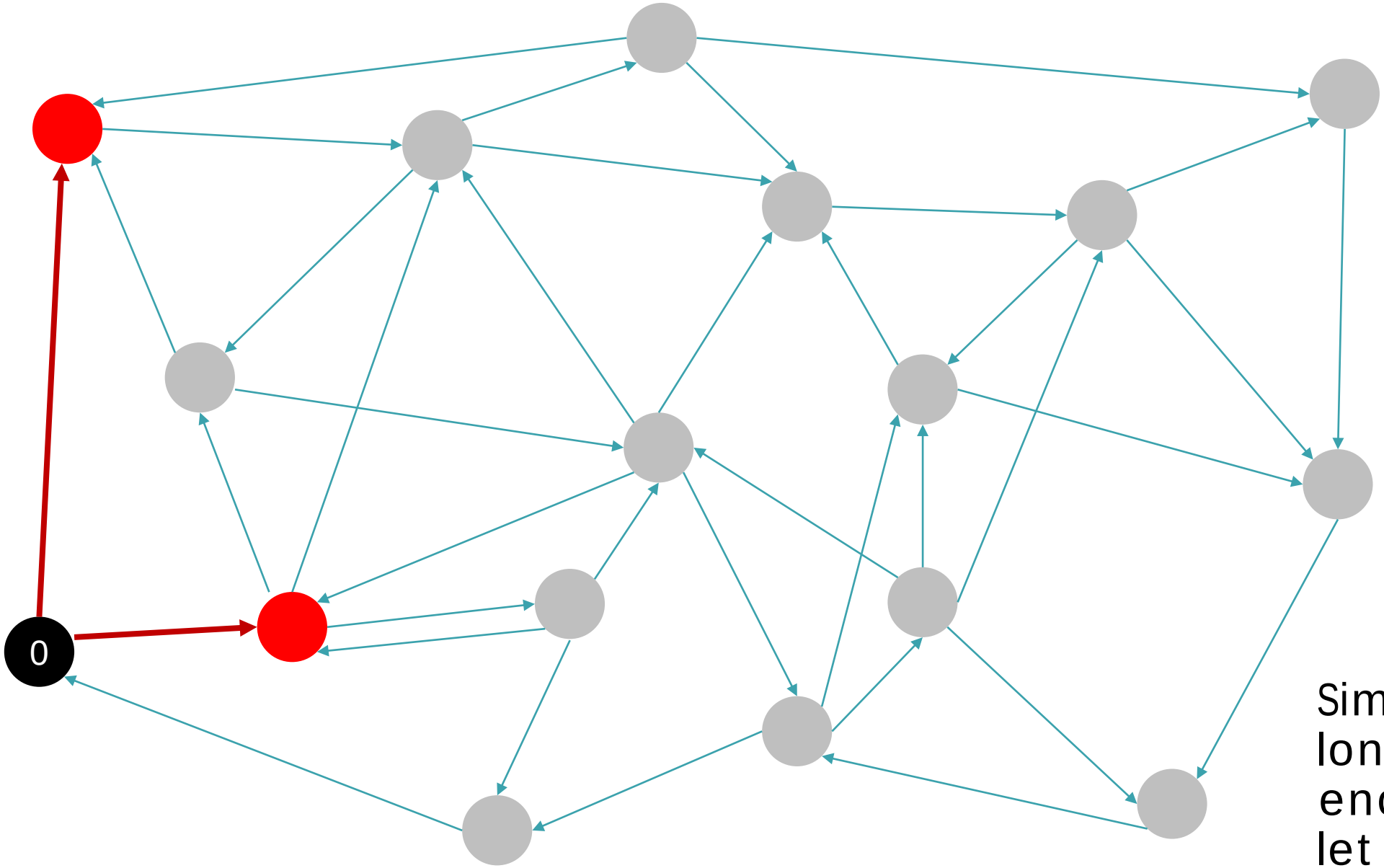
Single Source Shortest Path Embedding

- Graph G to Network N:
  - Vertex in G → Neuron in network N with 0 threshold and a long enough refractory period to make sure it fires only once
  - Edge in G → Synapse in network N with weight of 1 and delay proportional to the length of the edge in G
- Steps for shortest path algorithm:
  1. Stimulate source neuron
  2. Simulate for long enough a single spike travel along every edge in the network
  3. Read out spike times and updated graph
     1. Spike times of each neuron give the length of the shortest path from the source node to each destination node
     2. Potentiated edges give shortest paths

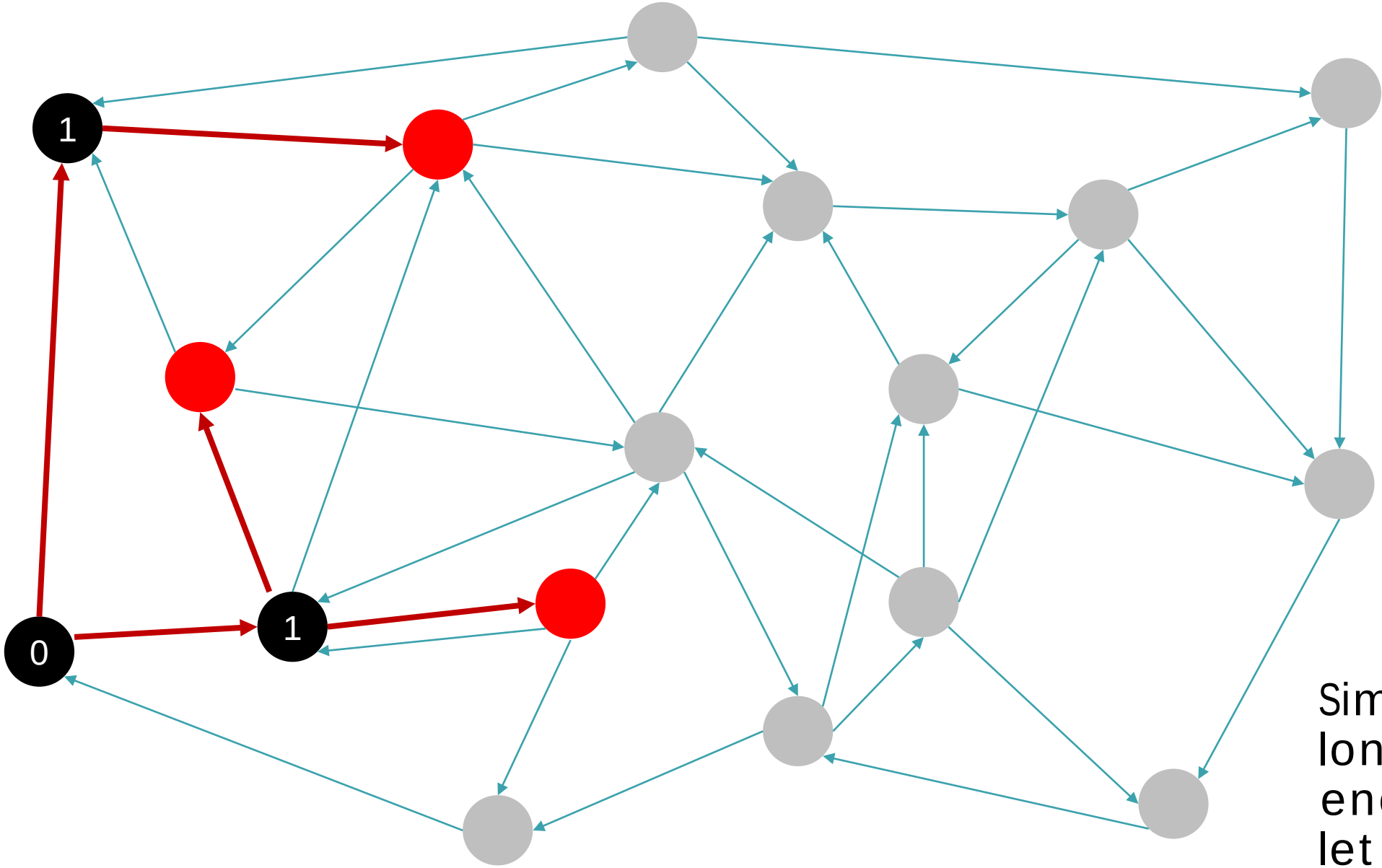




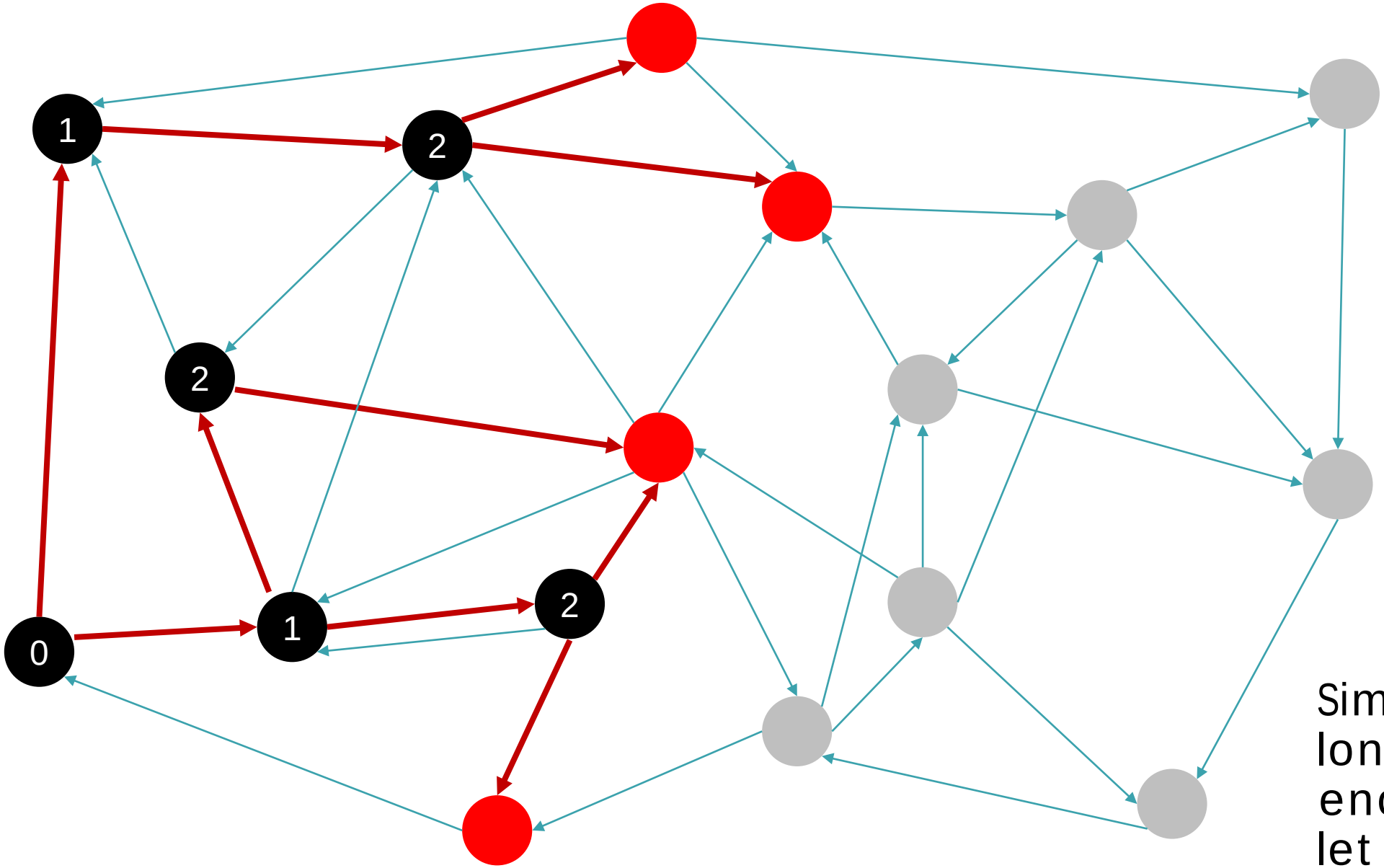
Stimulate source neuron



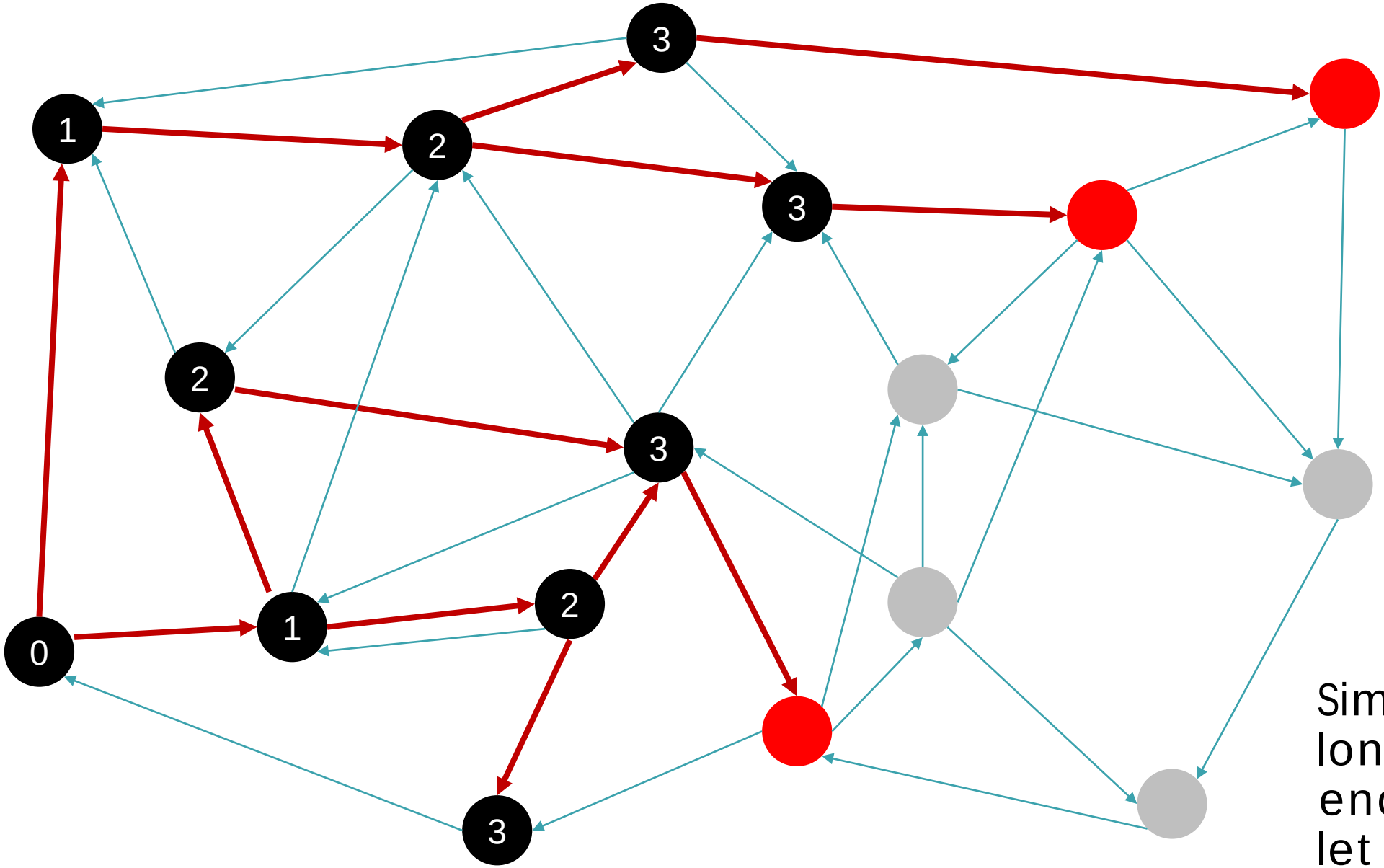
Simulate long enough to let spikes propagate



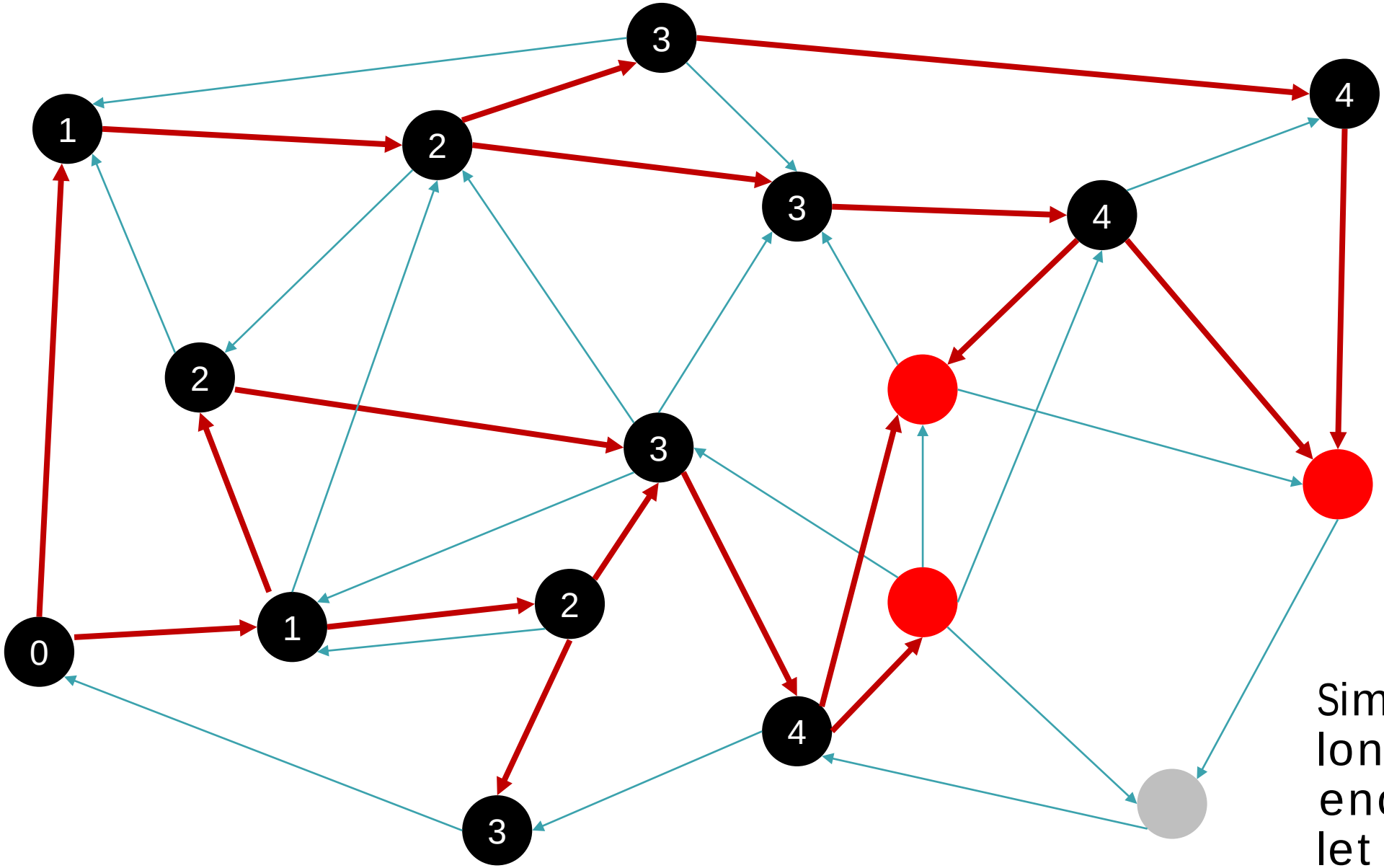
Simulate long enough to let spikes propagate



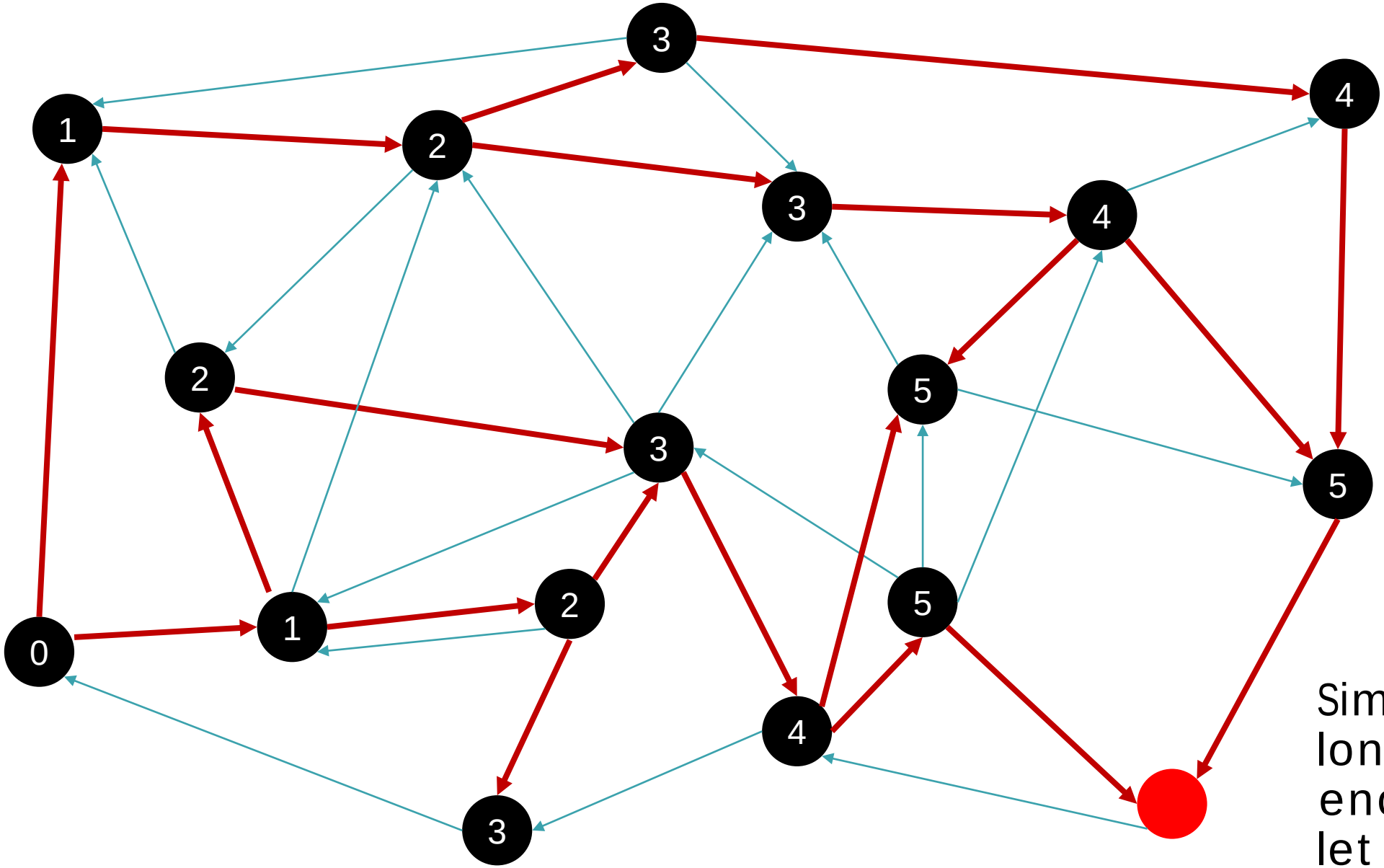
Simulate long enough to let spikes propagate



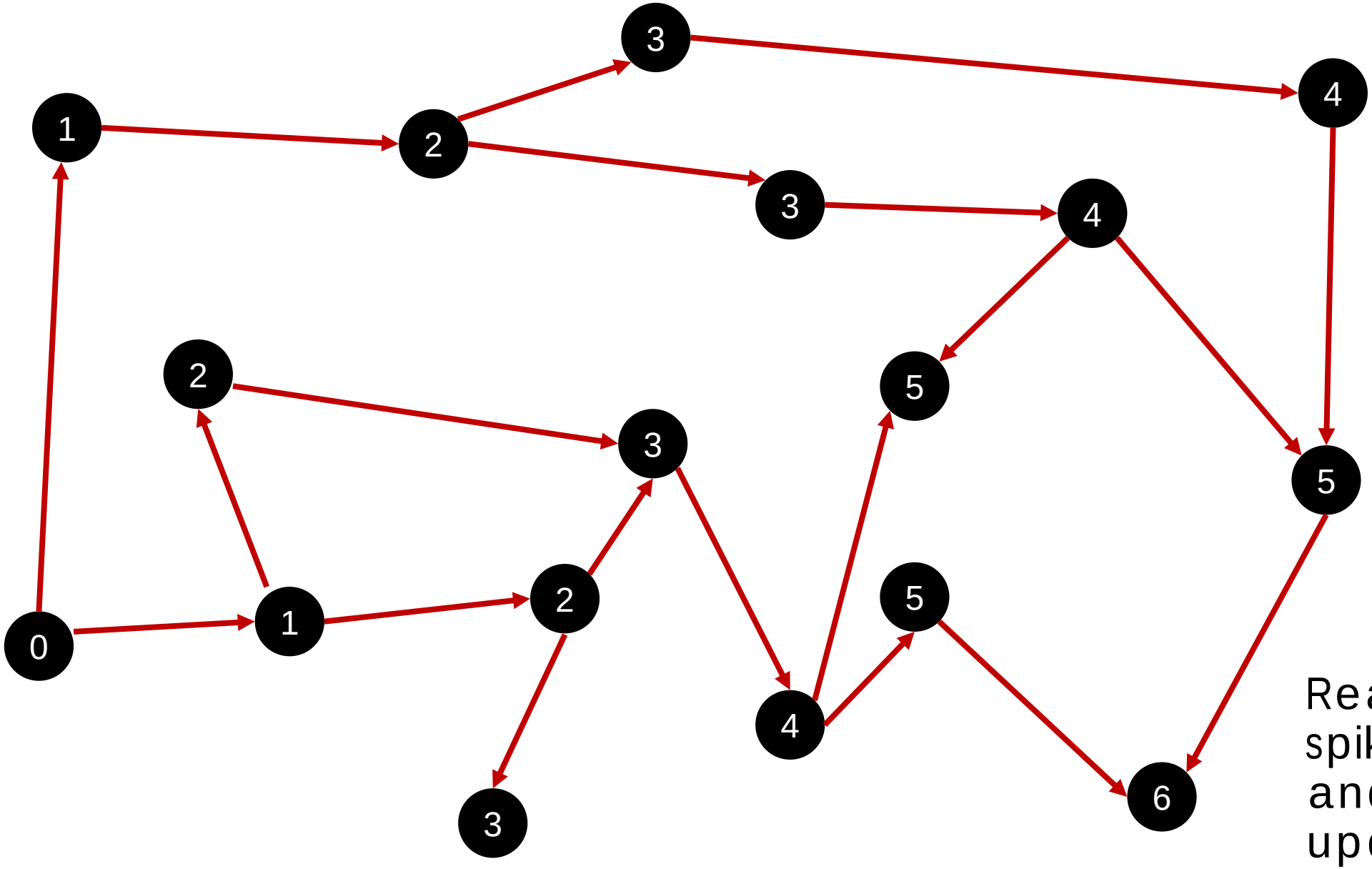
Simulate long enough to let spikes propagate



Simulate long enough to let spikes propagate



Simulate long enough to let spikes propagate



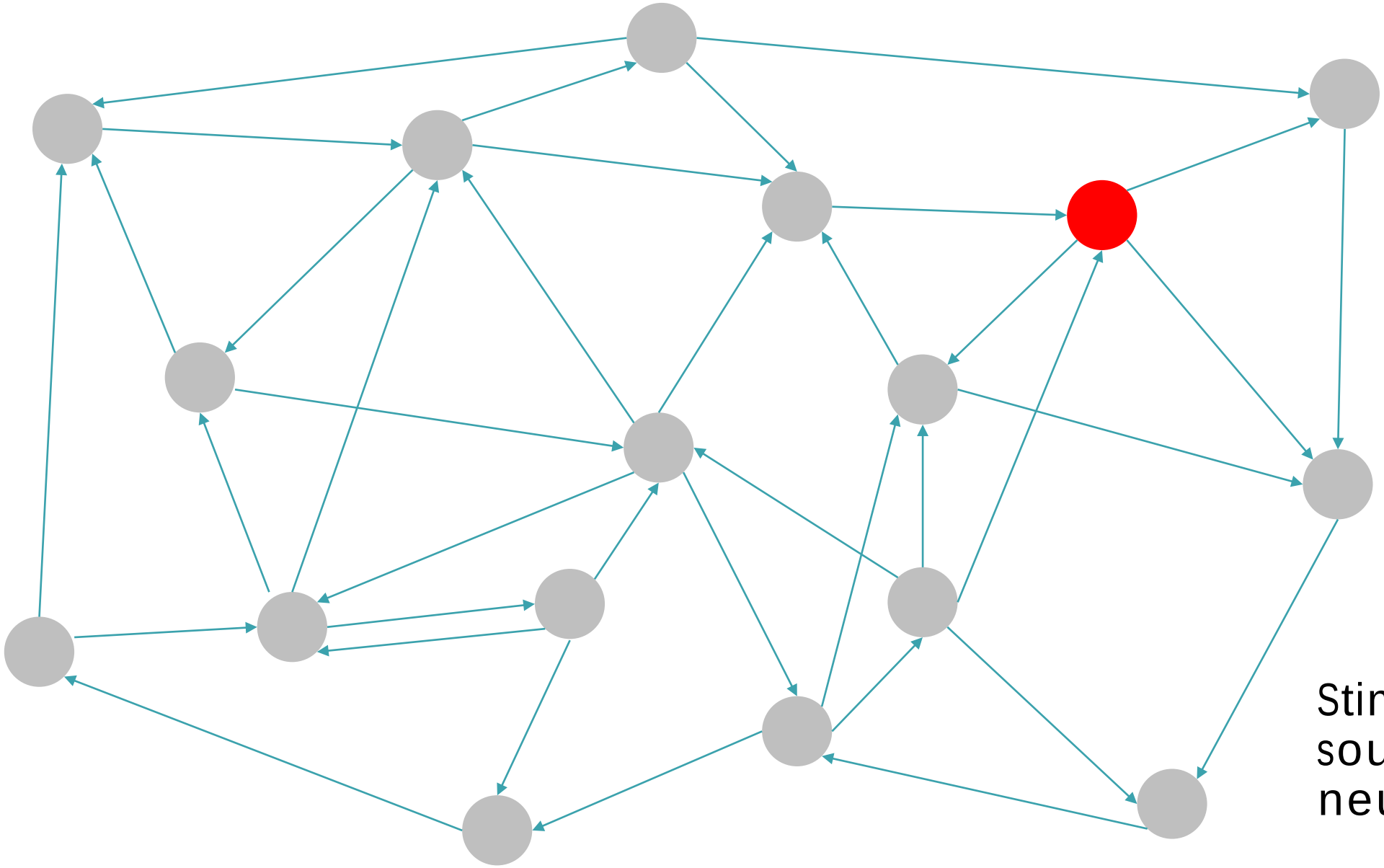
Read out spike times and updated graph

Neighborhood Subgraph Extraction

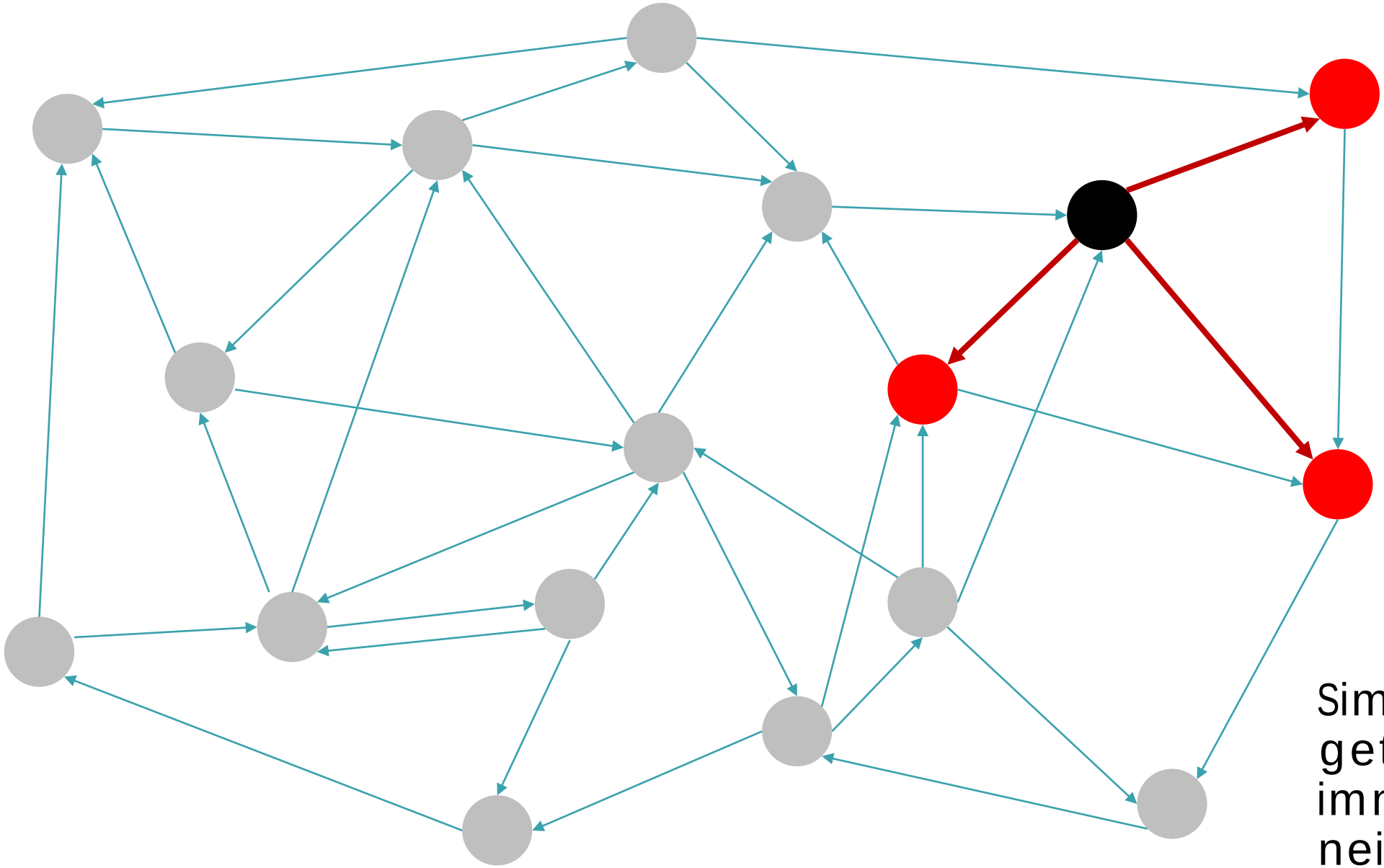
- Initial Graph G to Network N:
  - Vertex in G → Neuron in network N with threshold of 1 and refractory period of 1
  - Edge in G → Synapse in network N with weight of 1 and delay of 2

Neighborhood Subgraph Extraction

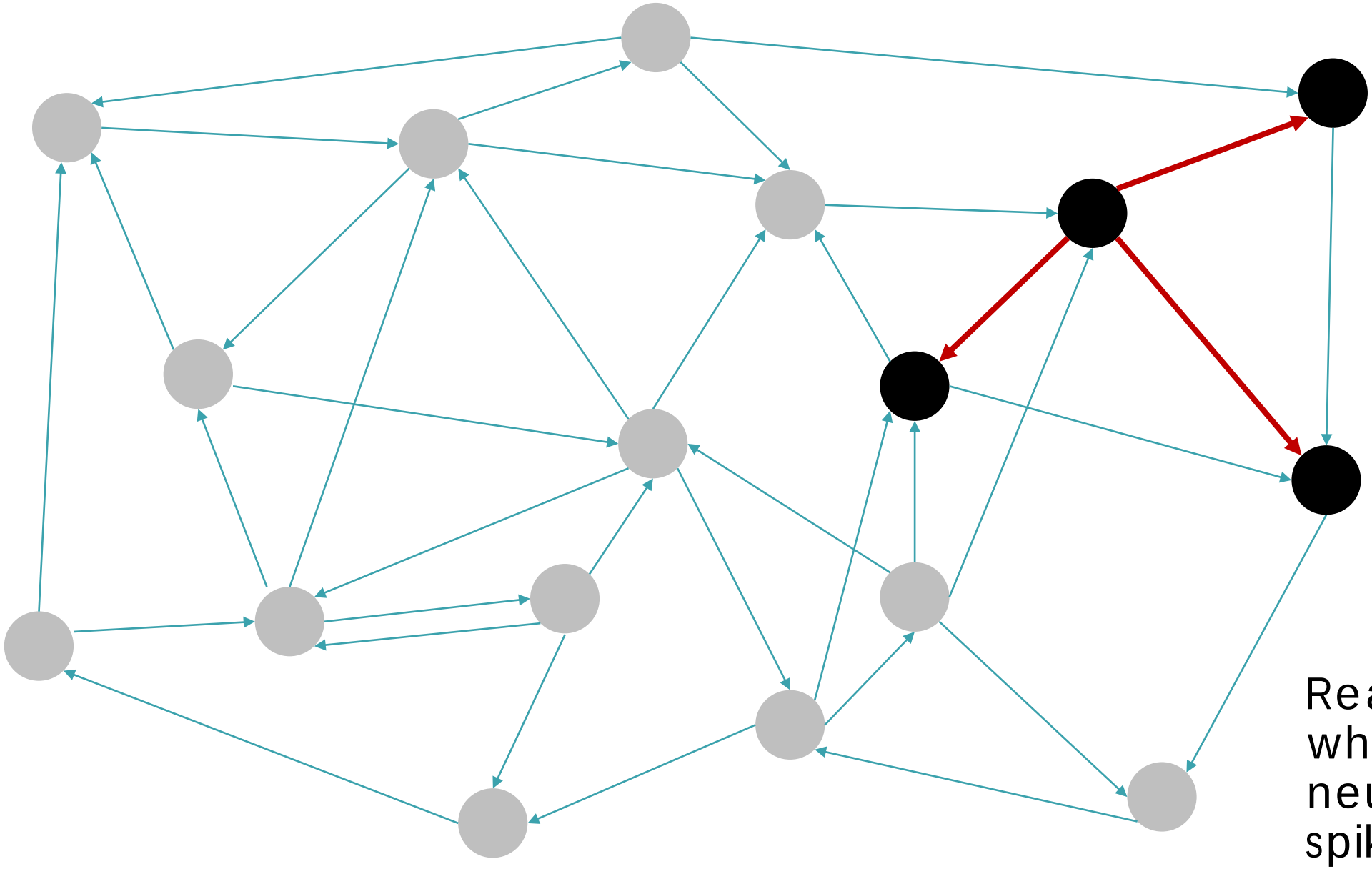
- Initial Graph G to Network N:
  - Vertex in G → Neuron in network N with threshold of 1 and refractory period of 1
  - Edge in G → Synapse in network N with weight of 1 and delay of 2
- Steps for neighborhood subgraph extraction
  1. Stimulate source neuron
  2. Simulate for two time steps
  3. Read out spike times to determine nodes in subnetwork N_S
  4. Update graph so that threshold of all other neurons outside of N_S is too high to fire
  5. Simultaneously fire all neurons in set N_S and simulate for two time steps.
  6. Read out graph to get potentiated edges, which are the edges for the subgraph



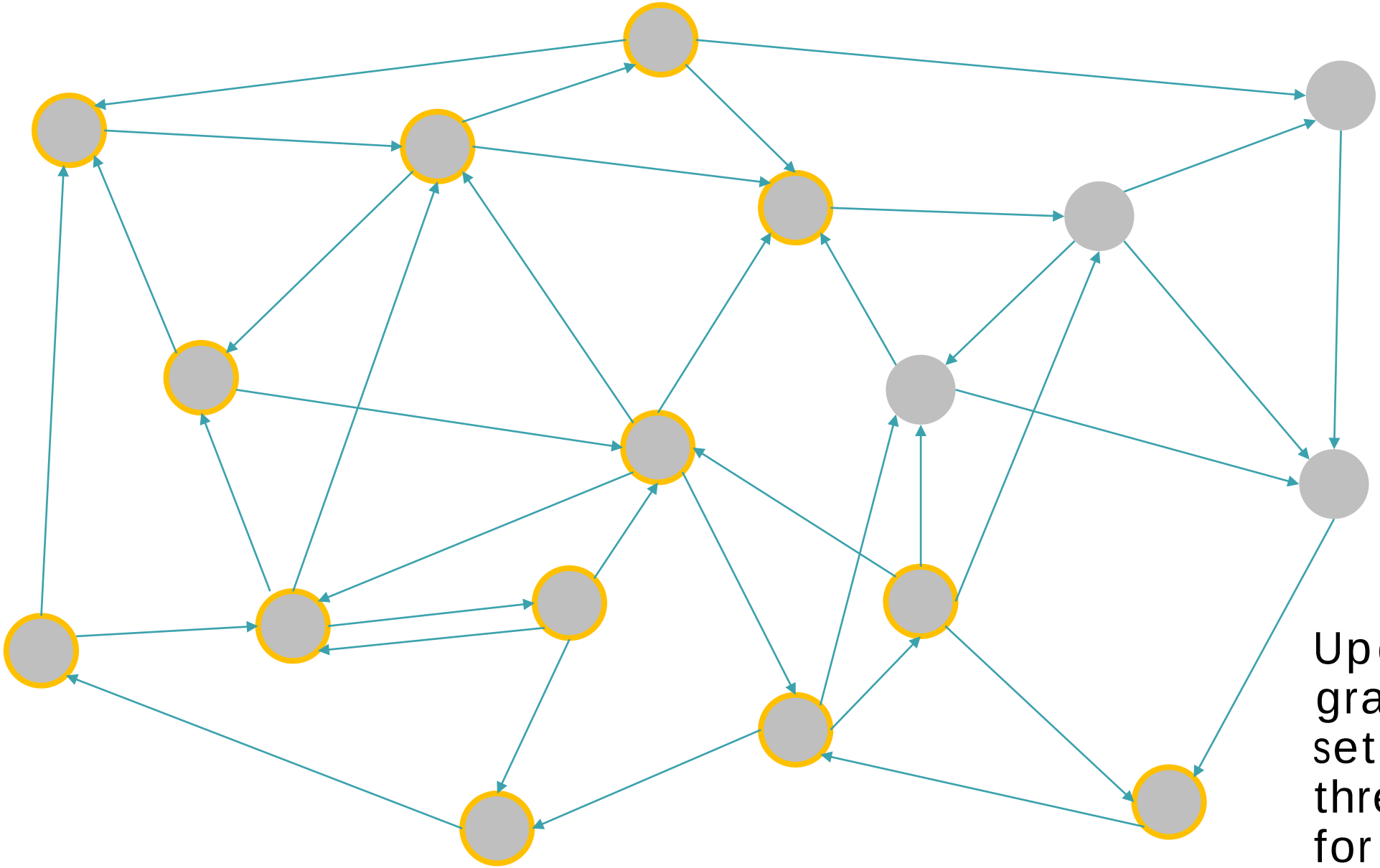
Stimulate source neuron



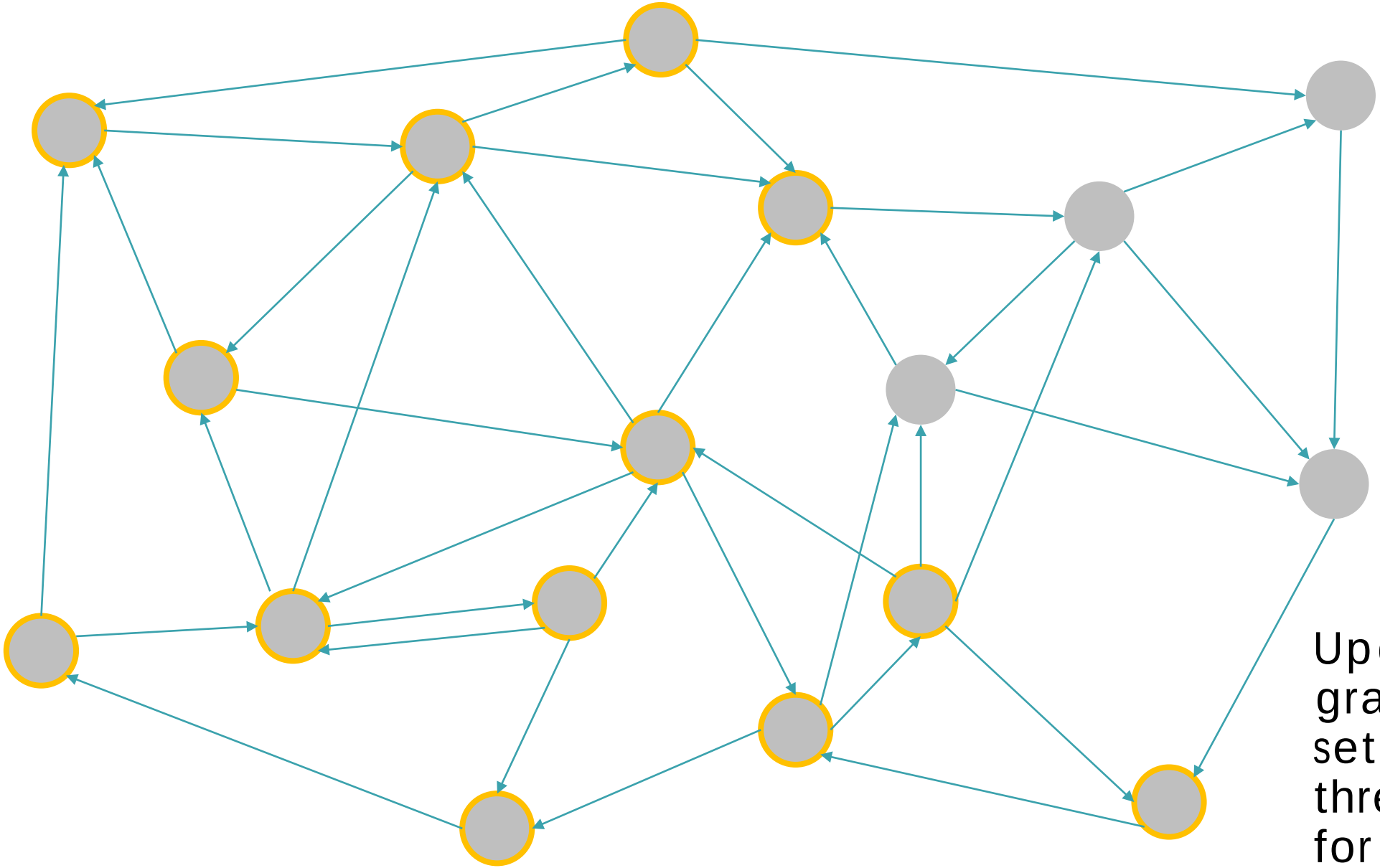
Simulate to get immediate neighbors



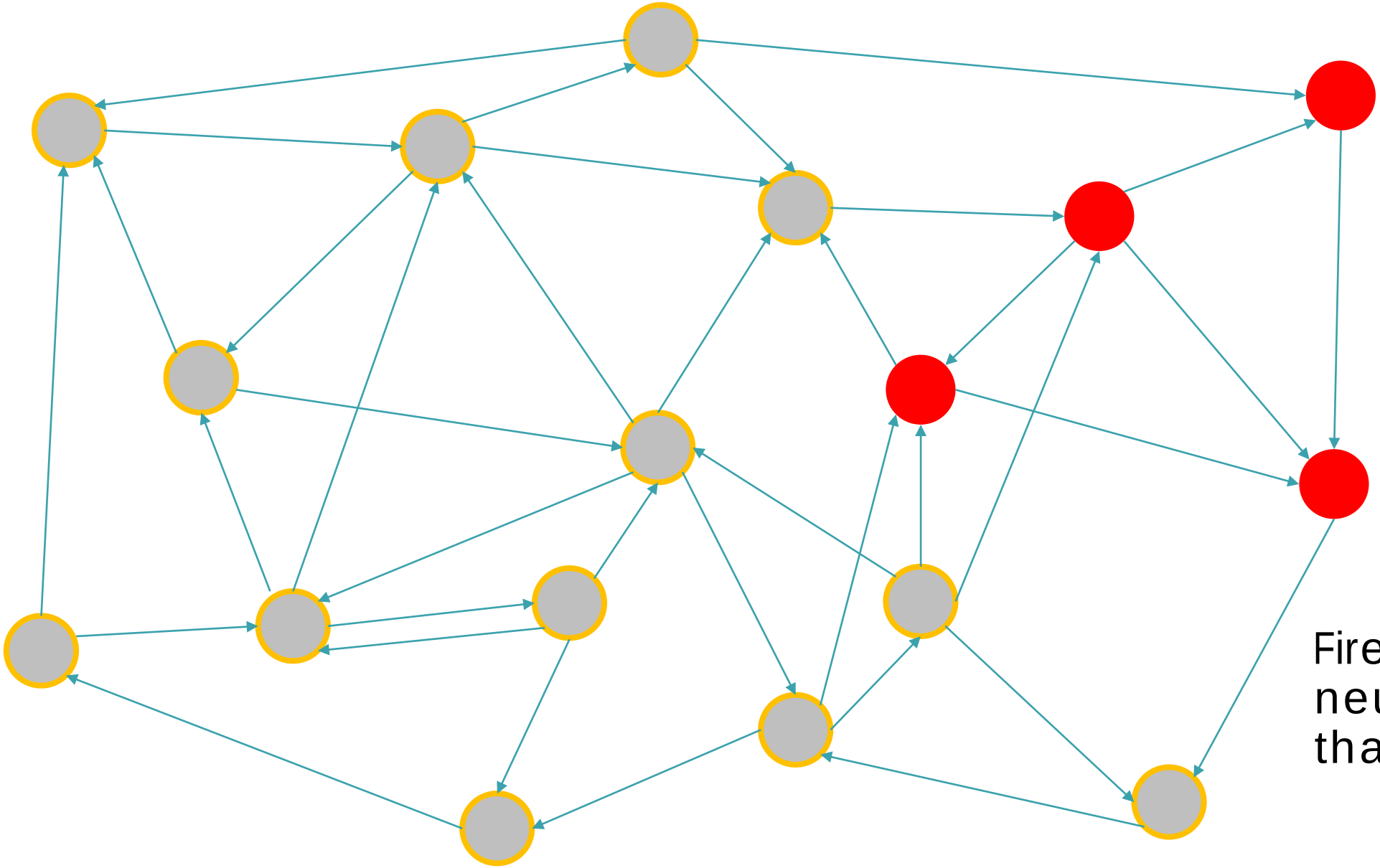
Read out which neurons spiked



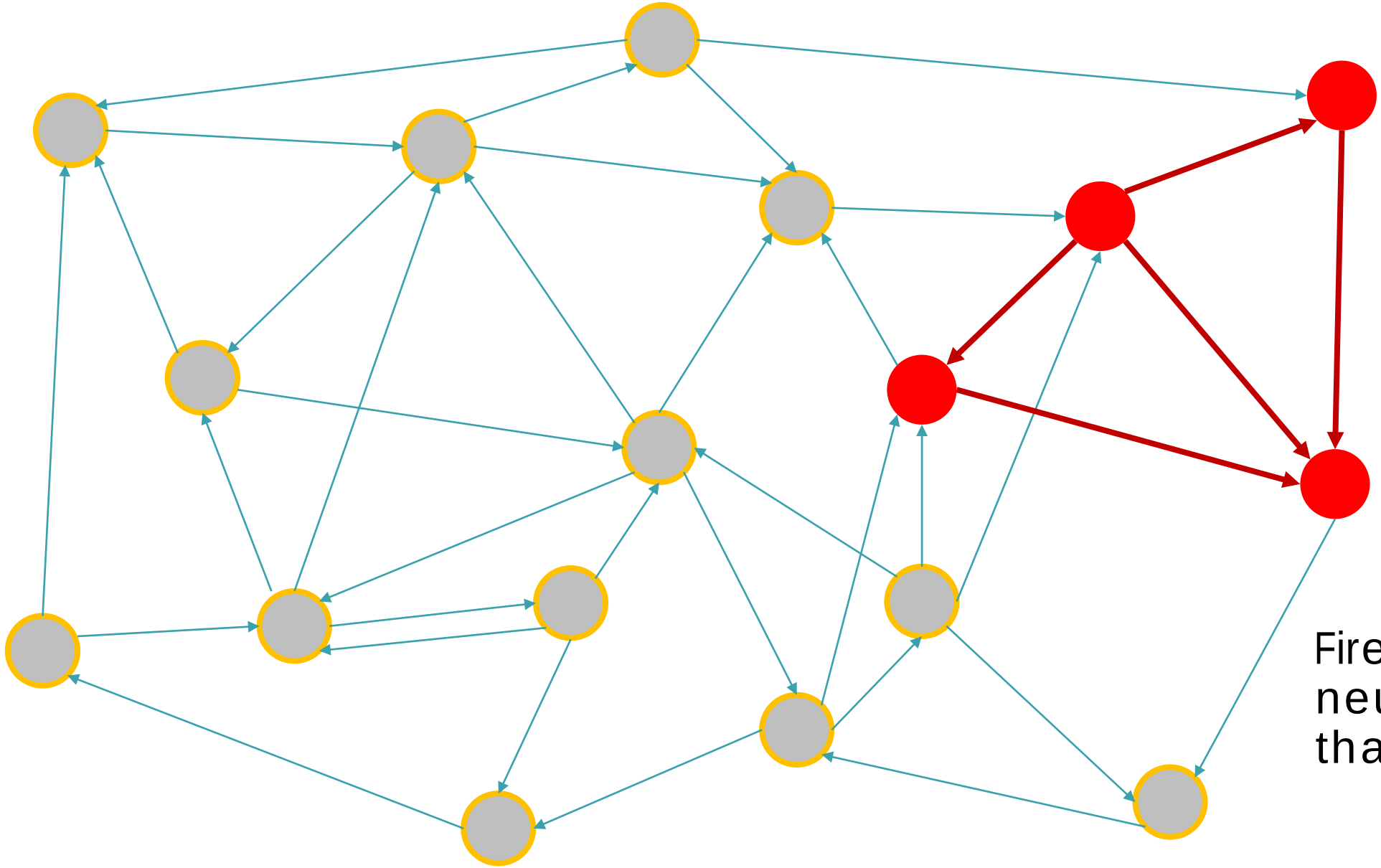
Update graph to set high thresholds for other neurons



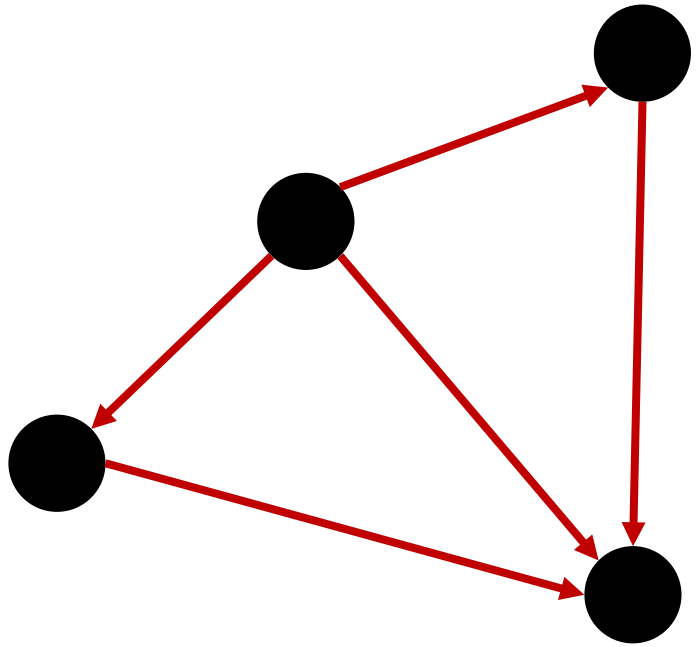
Update graph to set high thresholds for other neurons



Fire all neurons that spiked



Fire all neurons that spiked



Read out updated graph to find neighborhood edges

Neuromorphic/CPU Breakdown

	Shortest Path	Neighborhood Subgraph Extraction
Spiking Neuromorphic Co-Processor (SNC) Run Time	O(|E(G|)	O(1)
CPU Run Time	O(|E(G)|)	O(|V(G)|+|E(G)|)
CPU -> SNC Network Loads	1	2
SNC -> GPU Network Reads	1	1

Real-World Graphs

Graph	Type	Vertices	Edges
roadNet-CA	Undirected	1,965,206	2,766,607
Ca-HepPh	Undirected	12,008	118,521
Amazon0621	Directed	403,394	3,387,388

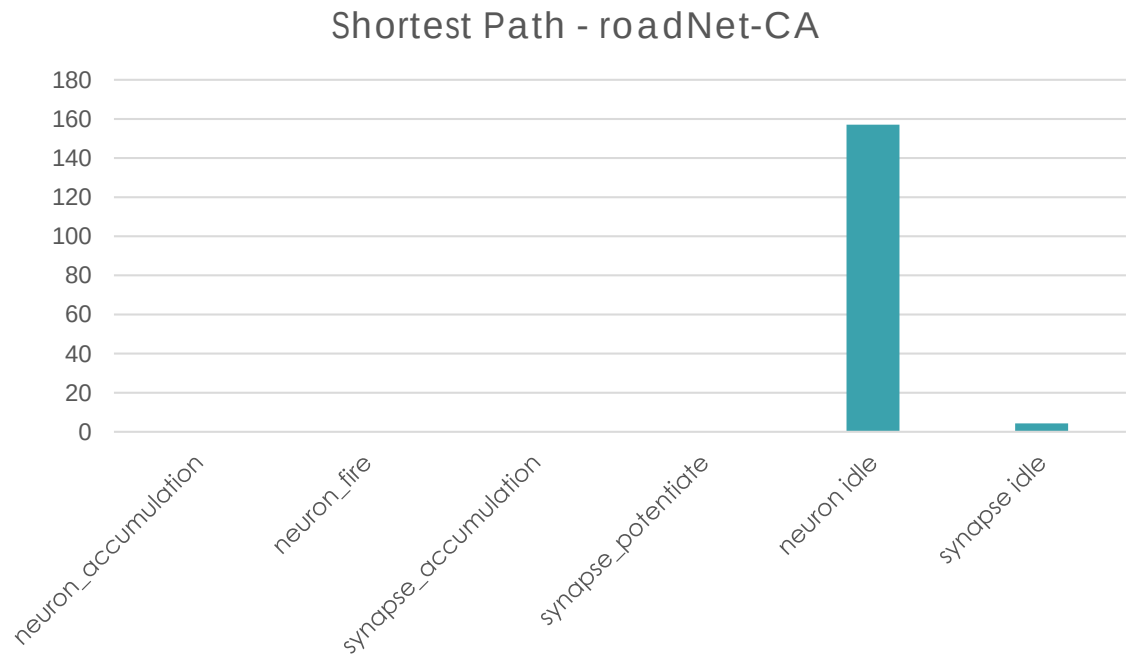
- roadNet-CA: Road network in California
- Ca-HepPh: Collaboration network graph from the high energy physics topic on Arxiv
- Amazon0621 – Amazon co-purchasing network from June 1, 2003

Energy Estimates for Each Graph

	roadNet-CA	ca-HepPh	amazon0601
Shortest Path	161.35 J	48.85 mJ	21.28 J
Neighborhood	58.32 µJ	999.14 nJ	12.56 µJ

Energy Estimates for Each Graph

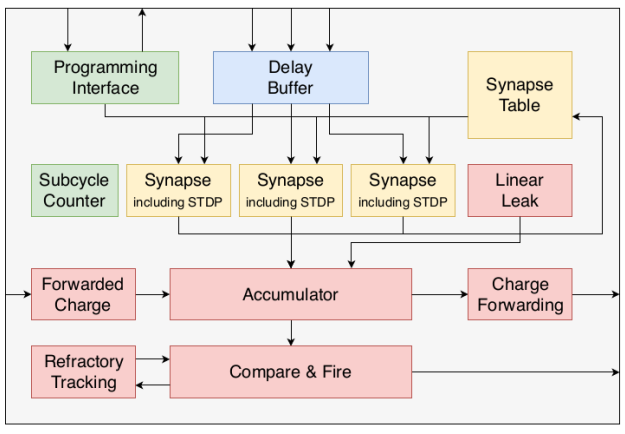
	roadNet-CA	ca-HepPh	amazon0601
Shortest Path	161.35 J	48.85 mJ	21.28 J
Neighborhood	58.32 µJ	999.14 nJ	12.56 µJ



What about other hardware?

Types of Neuromorphic Implementations

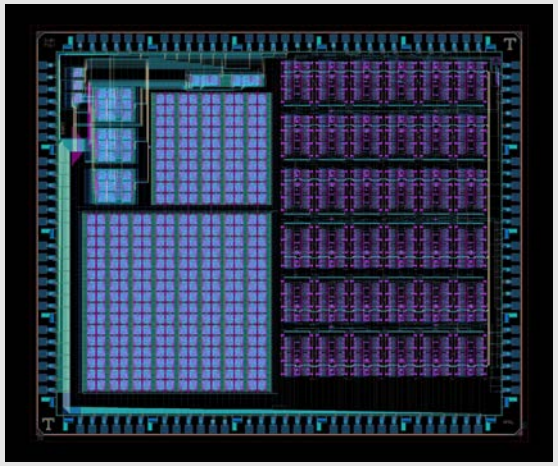
DANNA2



- Fully digital implementation
- Two versions:
  - DANNA2-dense is programmable
  - DANNA2-sparse is application-specific

Mitchell, J. Parker, et al. "DANNA 2: Dynamic adaptive neural network arrays." *Proceedings of the International Conference on Neuromorphic Systems*. ACM, 2018.

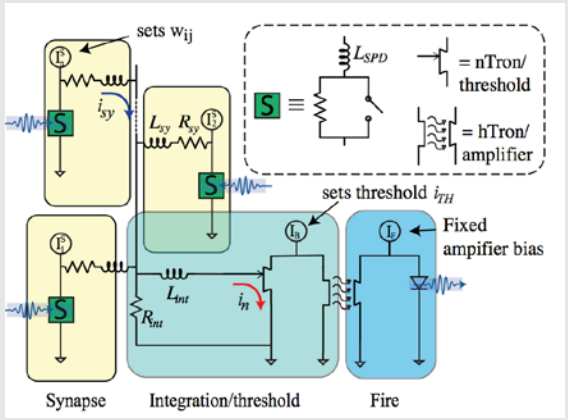
mrDANNA



- Mixed analog-digital implementation
- Synapses implemented with twin memristors
- Programmable

Chakma, Gangotree, et al. "Memristive mixed-signal neuromorphic systems: Energy-efficient learning at the circuit-level." *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 8.1 (2018): 125-136.

SOEN

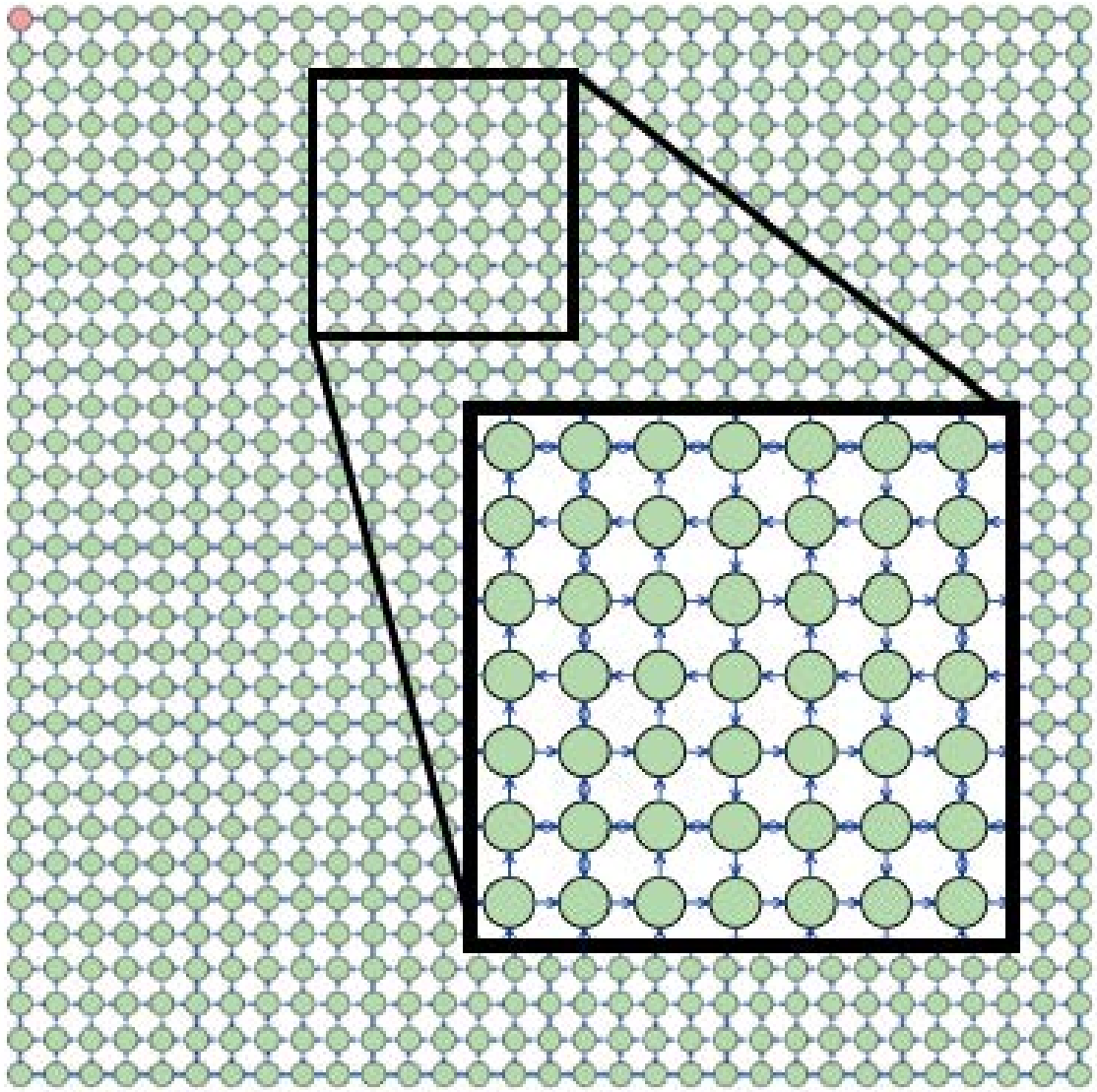


- Optoelectronic
- Neurons implemented using superconducting optoelectronics
- Delays are on neurons, not synapses

Buckley, Sonia, et al. "Design of superconducting optoelectronic networks for neuromorphic computing." *2018 IEEE International Conference on Rebooting Computing (ICRC)*. IEEE, 2018.

Another Example Benchmark Graph

- Example graph has 961 nodes and 2280 edges
- Graph is inspired by city-street layouts with some one-way streets and avenues and some two-way streets
- Find shortest path from source node to all other nodes



Shortest Path

Implementation	Neurons	Synapses	Energy
DANNA2-dense	961	2280	8.27 uJ
DANNA2-sparse	961	2280	1.47 uJ
mrDANNA	961	2280	1.84 uJ
SOEN	3242	4560	0.0599 uJ

Summary

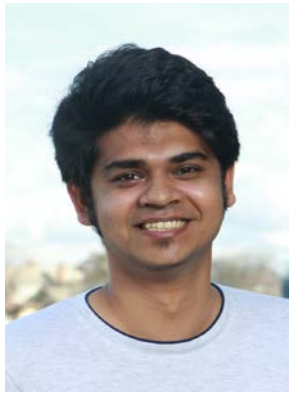
- Mapping non-machine learning tasks to neuromorphic systems is non-trivial, even for graph algorithms where the mapping is relatively natural.
- Different tasks can have radically different performance profiles.
- The goal of this work was not to give a direct comparison of neuromorphic vs. CPU or GPU approaches to graph algorithms, but we intend to plan to investigate that in future work.

Kathleen Hamilton, Tiffany Mintz, and Catherine Schuman. “Spike-based primitives for graph algorithms.” Available on arXiv: https://arxiv.org/abs/1903.10574

Acknowledgements

Kathleen Hamilton

Tiffany Mintz



Md Musabbir Adnan

Garrett Rose

Bon Woong Kyu

Sung Kyu Lim

- This work is funded by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory.
- The research used resources of the Oak Ridge Leadership Computing Facility.

International Conference on Neuromorphic Systems

- Join us at ICONS!
- July 23-25, 2019
- Knoxville, Tennessee
- Submission deadline:
  - ***April 15, 2019***
    - Short and full papers
    - Posters/short talk submissions
    - Tutorials and workshops
    - Special sessions

ICONS
International Conference on Neuromorphic Computing Systems

https://ornlcda.github.io/icons2019

Thank you!

Email: schumancd@ornl.gov
Website: CatherineSchuman.com