The University of Manchester

Low-Power, Fault-Resilient Communications in a Million-Core Neural Processing Architecture

Dr. Javier Navaridas

Lecturer in Computer Architecture

MANCHESTER 1824

The University of Manchester

Talk outline

- SpiNNaker Architecture
- Spiking neural networks
- Analysis of the interconnect
  - Multicast route generation
  - Stability under failures
  - Congestion sensitivity
  - Resilient to burstiness
- Conclusions

MANCHESTER 1824

The University of Manchester

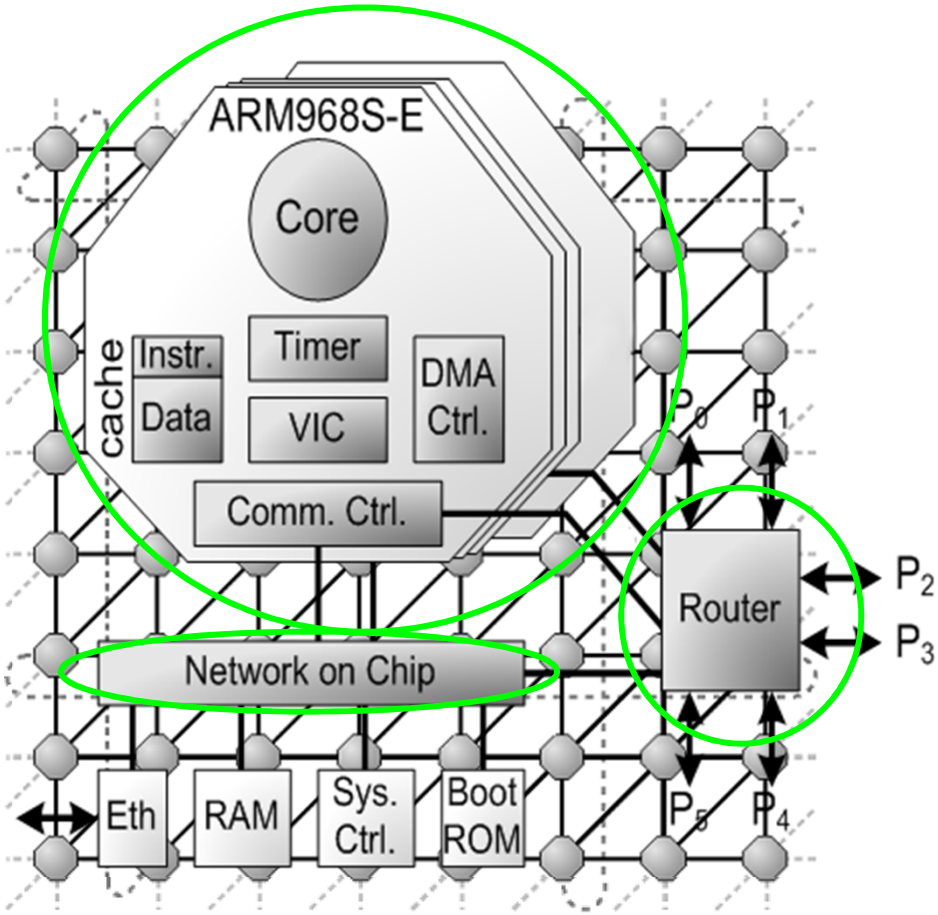
Overview of SpiNNaker

- Biologically-inspired massively parallel architecture
  - Up to 64K custom-made SoC chips
  - Over 1 million processors
- Simulation of very large-scale spiking neural networks in biological real-time
  - 10^9 neurons (~1% of the human brain)
  - 10^{12} synapses
- Communication-intensive application
  - Neuron activations are communicated to 1000s of other neurons
  - Up to 10^{13} communication events per second

Best architecture for supporting such a communication-demanding application ???

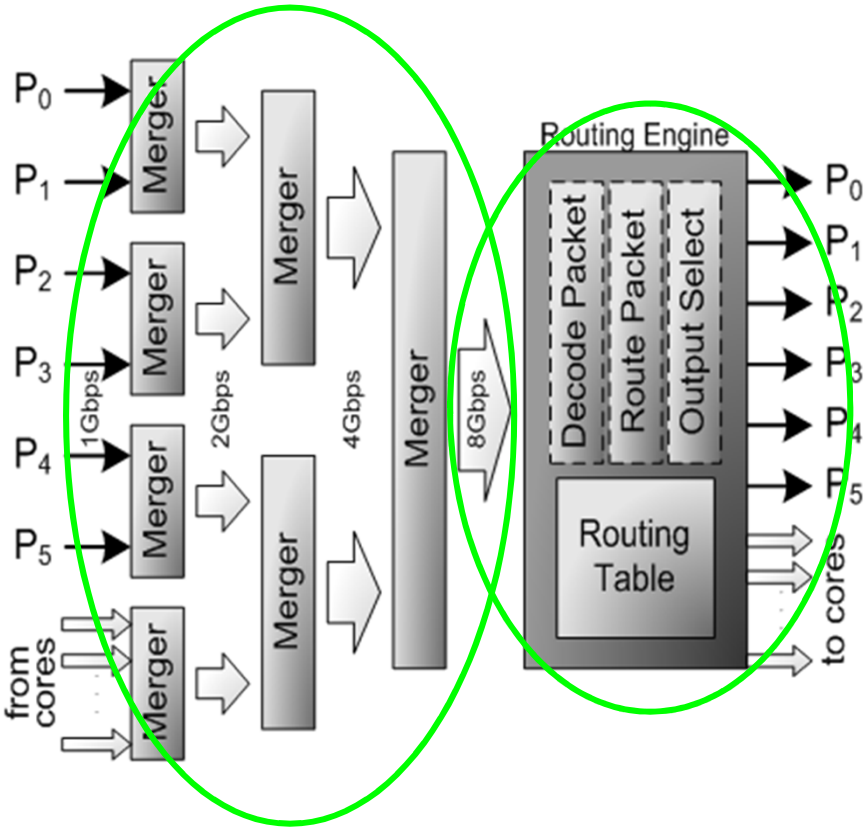
SpiNNaker architecture

- 18 extremely low power ARM cores
  - Up to 1000 neurons/core
  - ~1Watt per chip
- Asynchronous NoC
  - Connects chip resources
- Router-centric design
  - All packets pass through the router
  - Router accessed via the NoC
- Triangular torus topology
  - Tradeoff between redundancy, performance and scalability



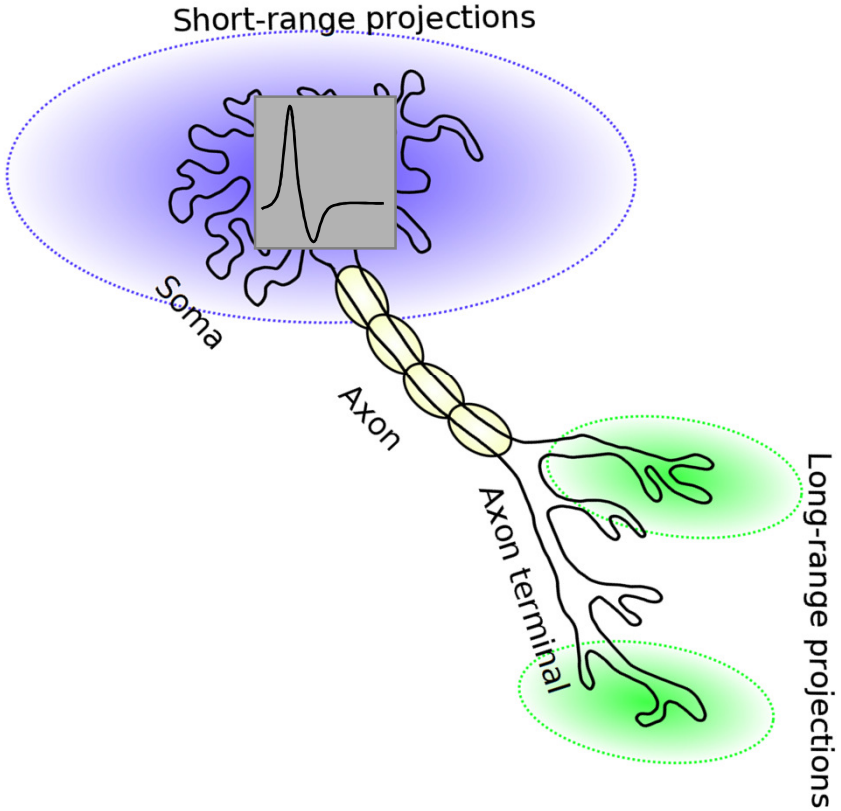
Router Architecture

- Hierarchical design
  - 3-stage port merging
    - Bandwidth doubled each stage
  - Packets use the routing engine once at a time
    - Router ~30x faster than links
    - Not expected to become the bottleneck
- Routing engine
  - Native Multicast
  - Associative routing tables (Source-based routing)
  - Built-in Emergency routing mechanism to exploit topological redundancy



Application Model

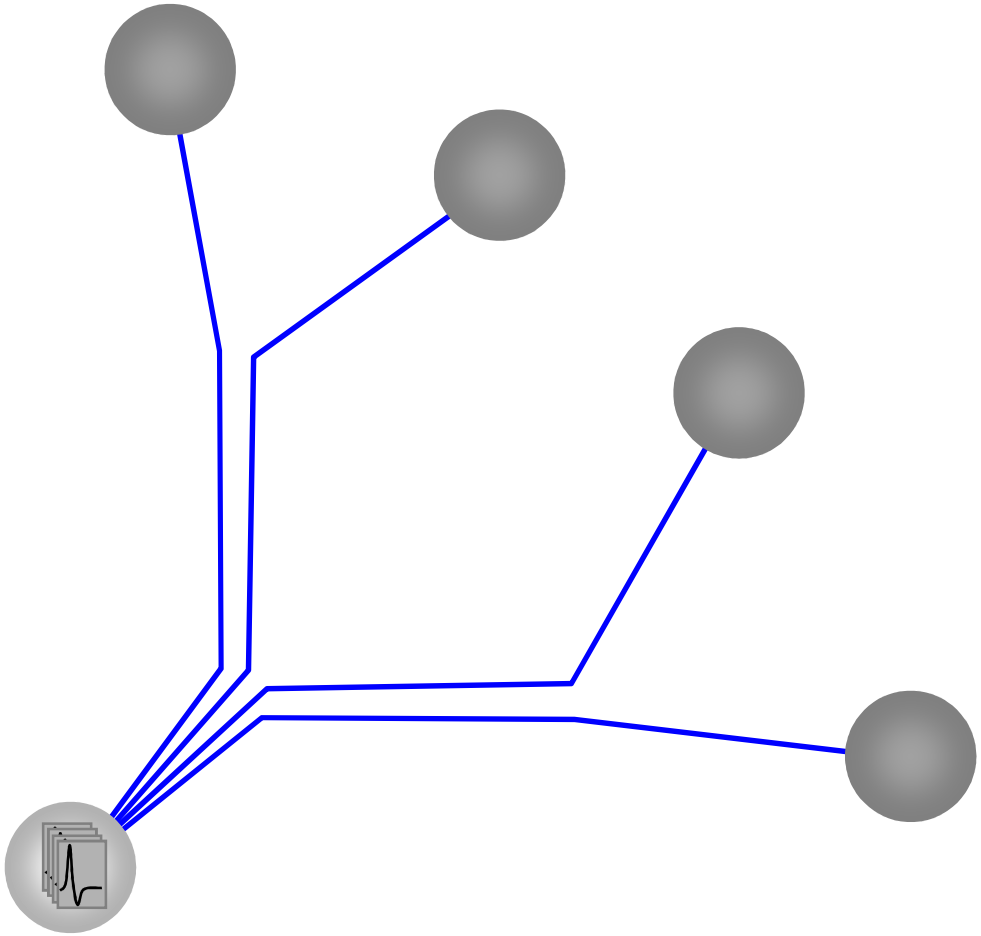
- PyNN is used to define neural nets
  - Highly integrated frontend
- Neurons stimulated beyond a threshold communicate through spike events
  - Biological neuronal nets favour local connectivity
  - These spikes need to be delivered to large sets of destinations (~1000)
  - Neuron multiplexing and multicast interconnects make this problem approachable
- How to deal with this high fan-out?



MANCHESTER 1824

The University of Manchester

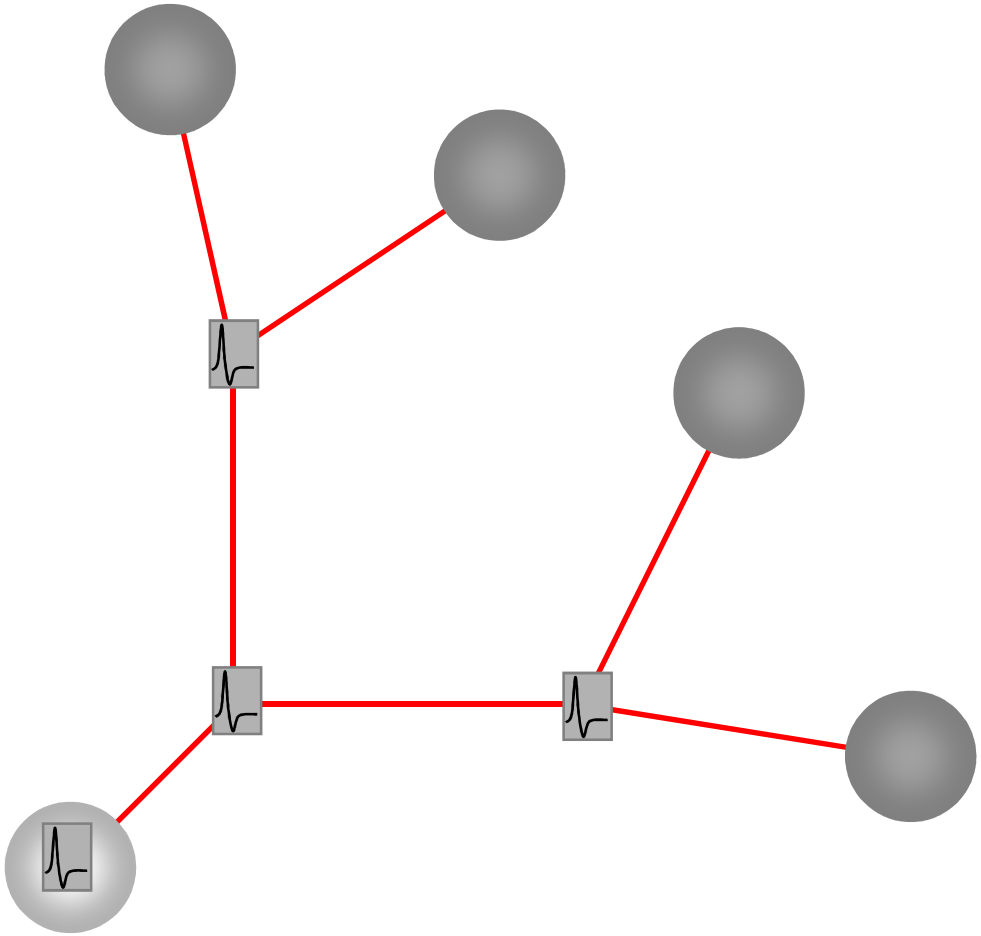
Unicast Delivery



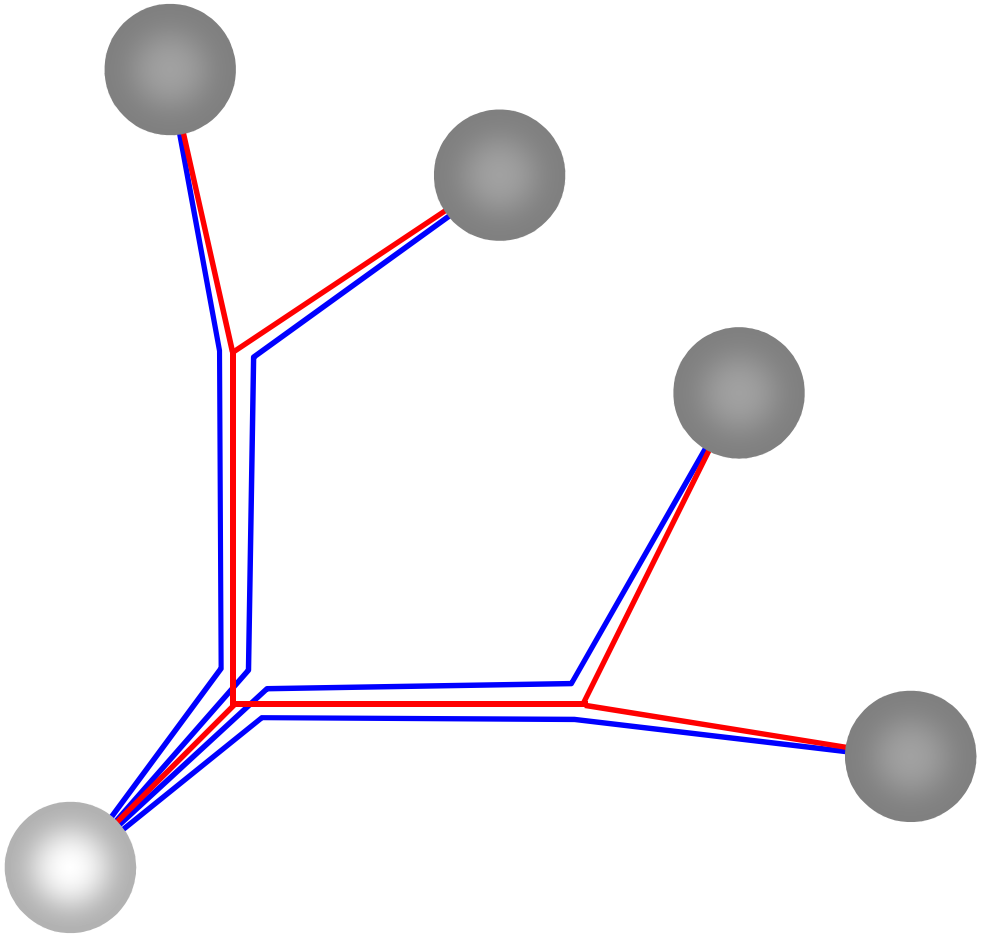
MANCHESTER 1824

The University of Manchester

Multicast Delivery



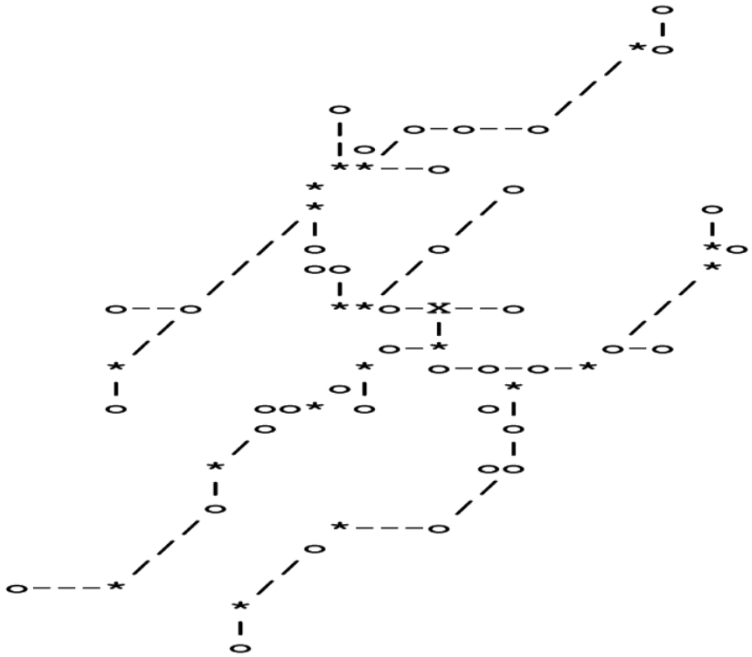
Unicast *vs* Multicast



- Multicast is more efficient
  - Avoids saturating links near the source
  - The overall bandwidth consumed is much lower
    - (7 vs 12 in this simple example, but unicast does not scale well)
- Generating multicast routes is far from trivial
  - We explored strategies for generate these routes

Multicast Routing Strategies

- Simple Oblivious Shortest Path Strategies
  - Dimension Order Routing
    - Baseline strategy, commonly used
    - Traverse the dimensions in a given order (XYD, in our case)
  - Longest Dimension First Routing
    - Traverse the dimension with more hops, then the other
- Smart Exploration Strategies
  - Enhanced Shortest Path Routing
    - Adapts routes only within the shortest path
  - Neighbourhood Exploring Routing
    - Explores in all direction and not restricted to shortest paths

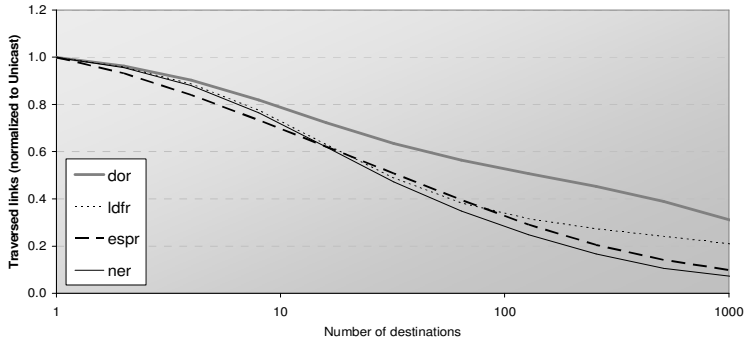


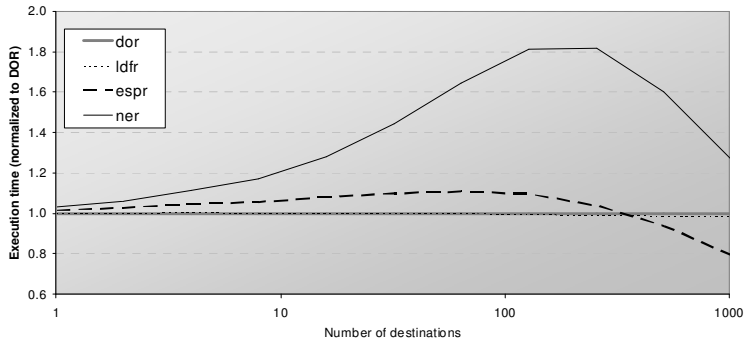
MANCHESTER 1824

The University of Manchester

Multicast Routing Strategies

- Multicast is better suited for large-scale simulation of neural nets than unicast
  - Up to 10x less network bandwidth needed
- We proposed a well balanced collection of multicast route generation strategies
  - NER can be used to reduce network bandwidth
  - LDFR can be used to generate routes faster
  - ESPR offers a tradeoff between the two
  - DOR, the first strategy used by most groups in the community is not very well suited for multicast routing
  - Work on adapting them for fault-tolerance





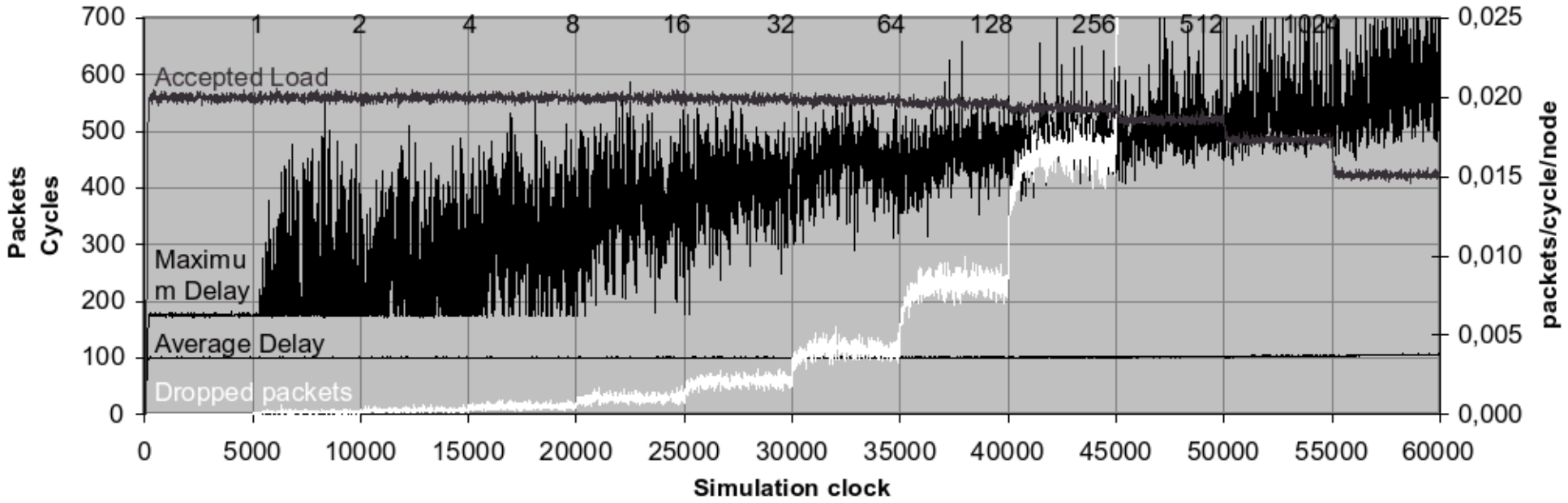
MANCHESTER 1824

The University of Manchester

System Stability under Failures

*Emergency Routing **NOT** Activated*

- Very unstable behavior
  - Maximum delay heavily fluctuates
  - Dropped packets grows linearly with failures
  - Accepted load drops



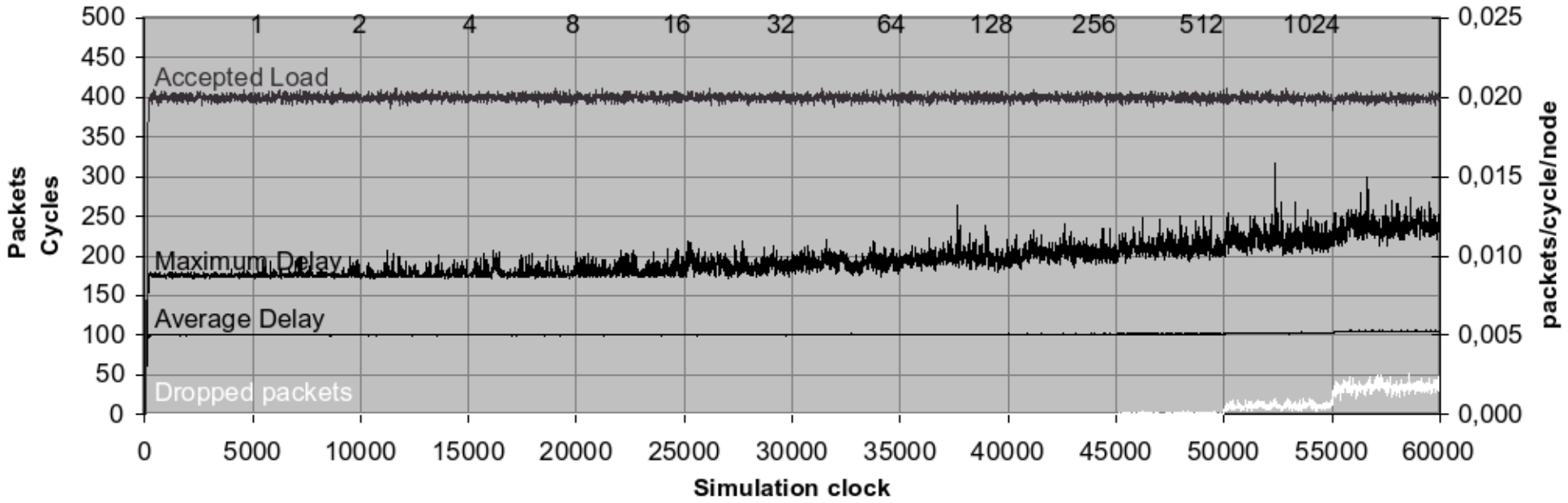
MANCHESTER 1824

The University of Manchester

System Stability under Failures

Emergency Routing Activated

- Quite stable behavior
  - Maximum delay experiences slightly varies
  - Packets dropped remain 0 until 128 failures
- Shows Emergency routing suitability

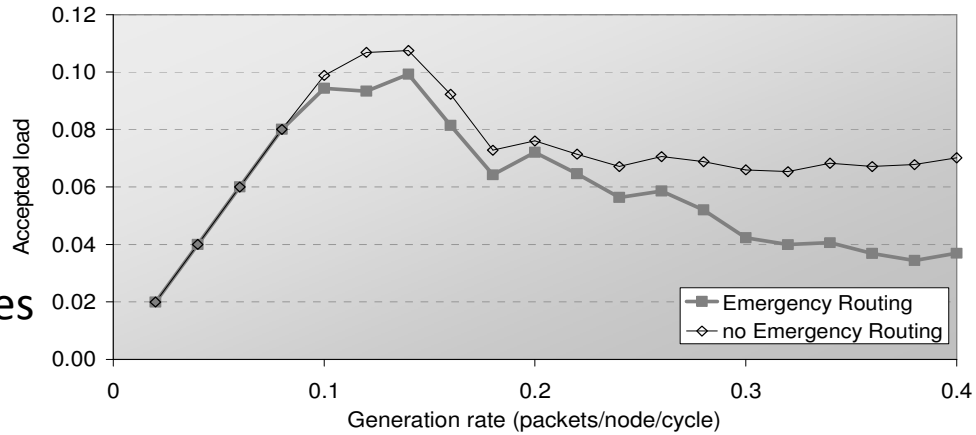


MANCHESTER 1824

The University of Manchester

Effects of network congestion

- Emergency routing is known to be highly sensitive to congestion
  - Throughput before saturation (typical scenario) is not affected
  - Throughput beyond saturation is significantly reduced (up to 50%)
  - Emergency routed packets create more pressure in the reroute path, which exacerbates congestion issues



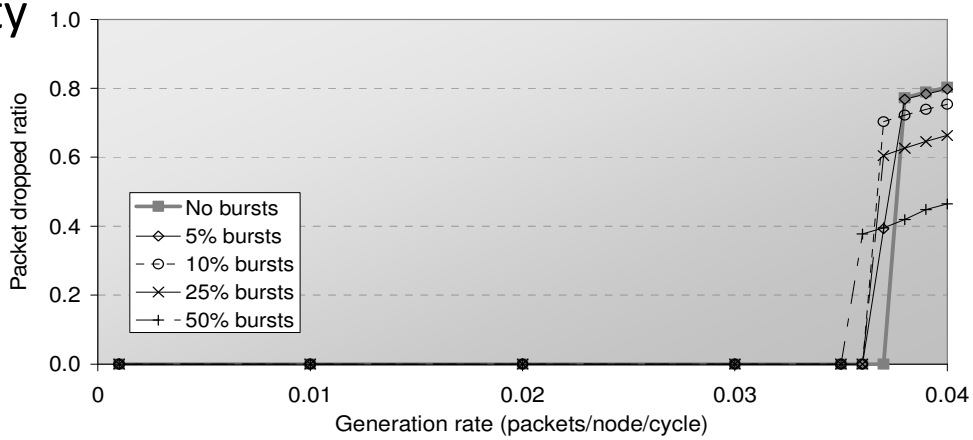
- Dynamic activation of Emergency routing when failures are detected
  - Avoids issues with transient congestion
  - Benefits from higher stability when failures in the network

MANCHESTER 1824

The University of Manchester

Effects of traffic Burstiness

- SpiNNaker's execution model exhibits bursty communication
- We analysed the interconnect ability to deal with such traffic model
  - Increasing the occurrence of long bursts barely affects the behaviour of the network
  - Network saturation point is slightly reduced (by ~5% for 50% bursts)
- In general the network behaves in a stable manner regardless of traffic burstiness



MANCHESTER 1824

The University of Manchester

Conclusions

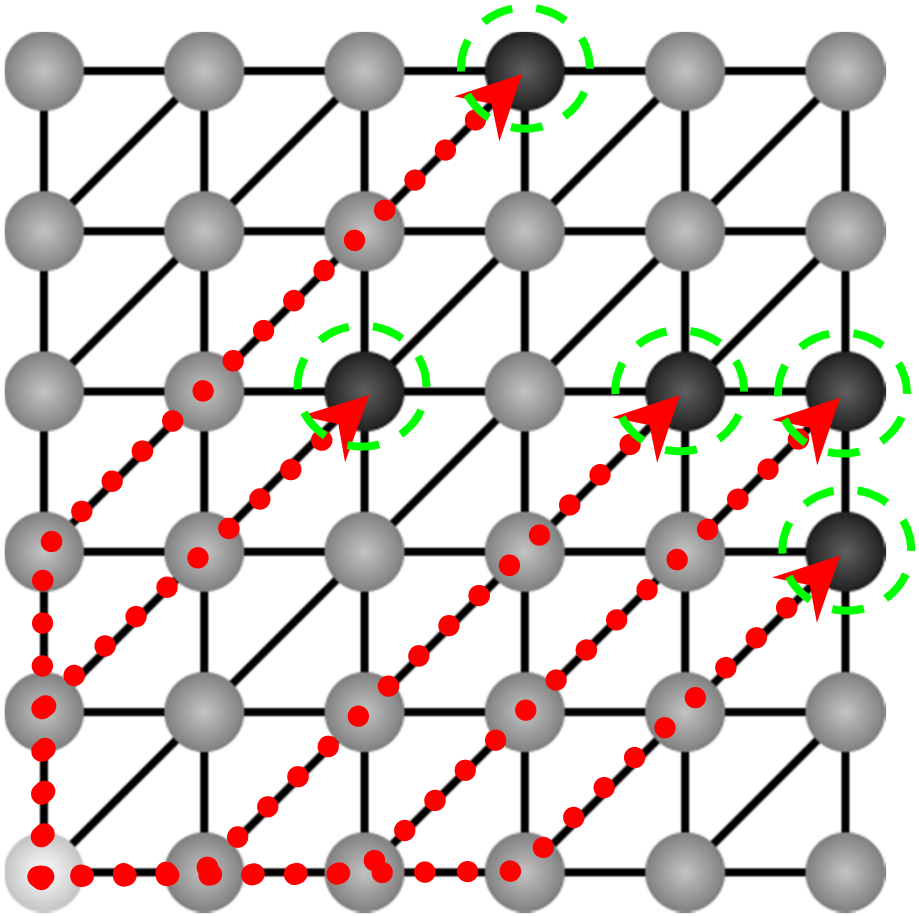
- Discussed SpiNNaker's most important features and strengths
  - Chip and interconnect architecture
  - Application Model
  - Full-fledged system construction progress
- SpiNNaker's low-spec, custom-made multicast router can sustain the communication-demanding neural applications
  - Stable in most foreseeable situations as regards traffic load, burstiness and network failures
  - Only in extreme scenarios of low locality may the network become a bottleneck
  - Multicast engine reduces network pressure
  - Provided a well-balanced collection of routing algorithms

Thanks for your attention

Questions

Extra slides

Dimension Order Routing

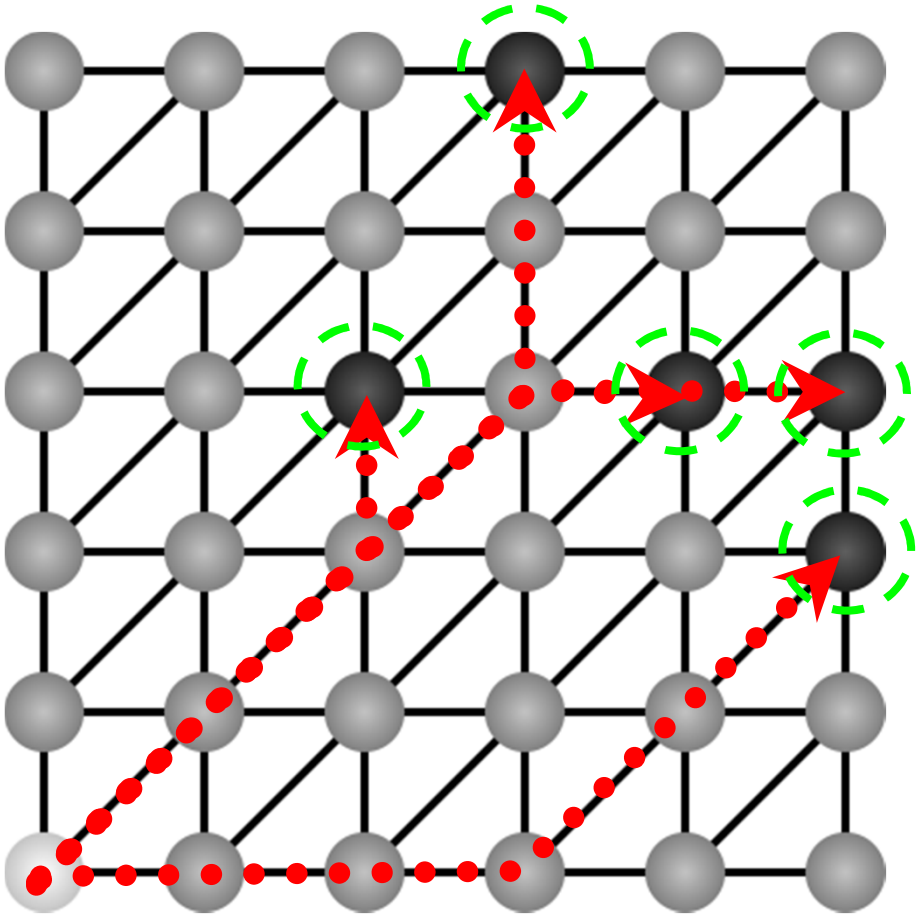


Lots of parallel branches – not very efficient

MANCHESTER 1824

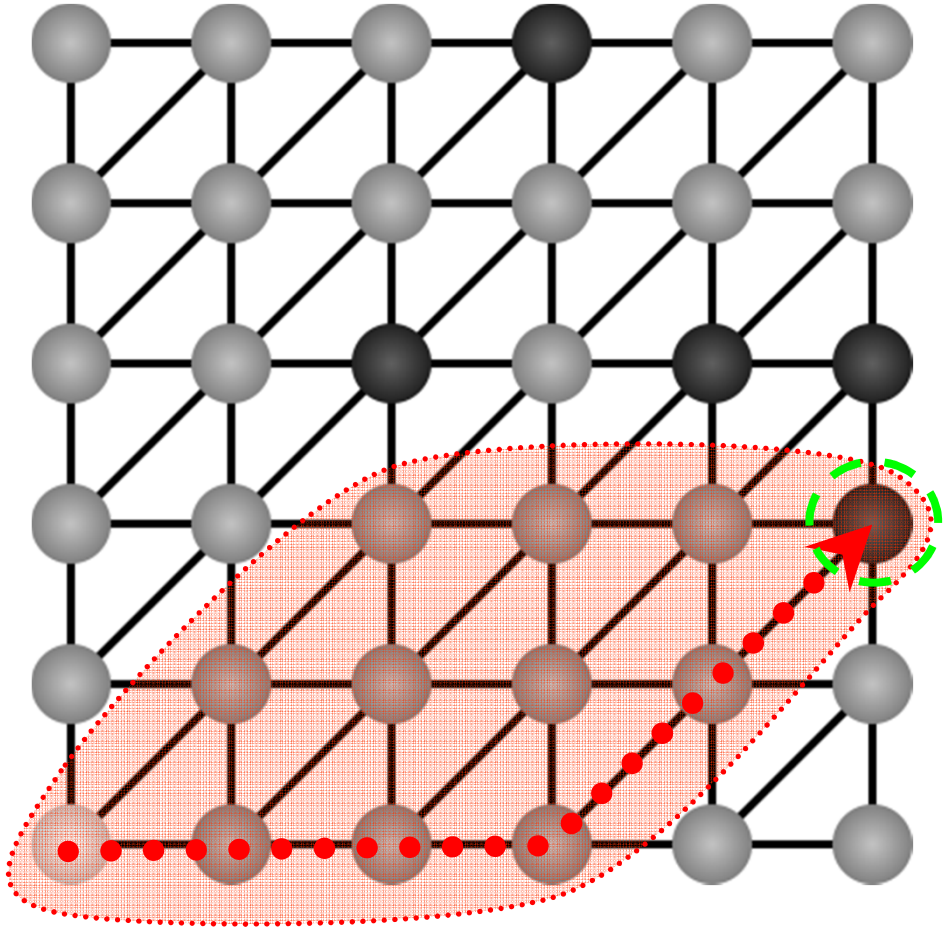
The University of Manchester

Longest Dimension First Routing



Main diagonal reused often

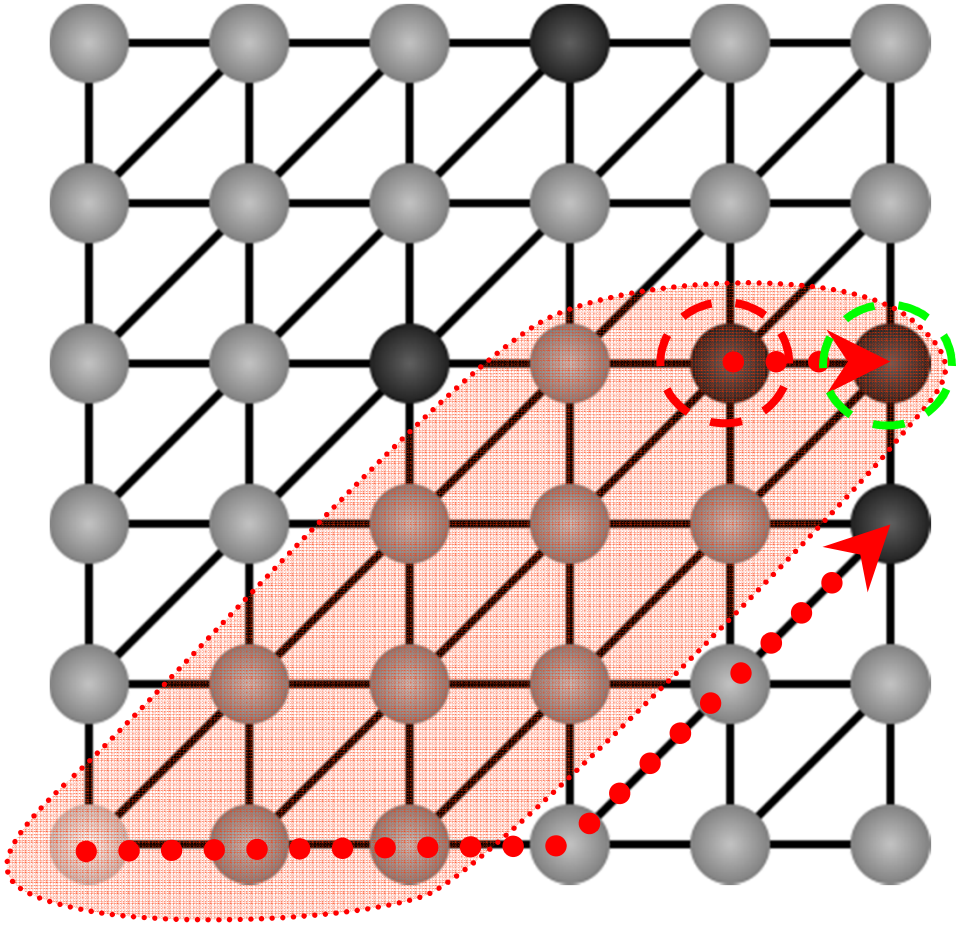
Enhanced Shortest Path Routing



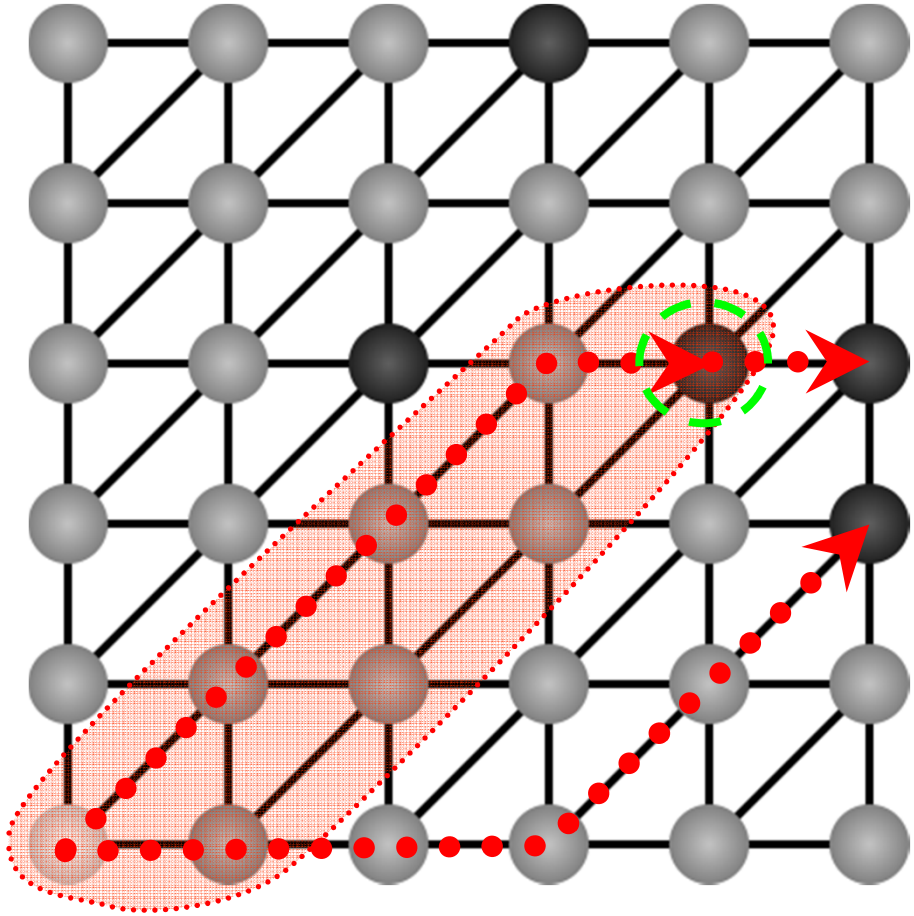
MANCHESTER 1824

The University of Manchester

Enhanced Shortest Path Routing



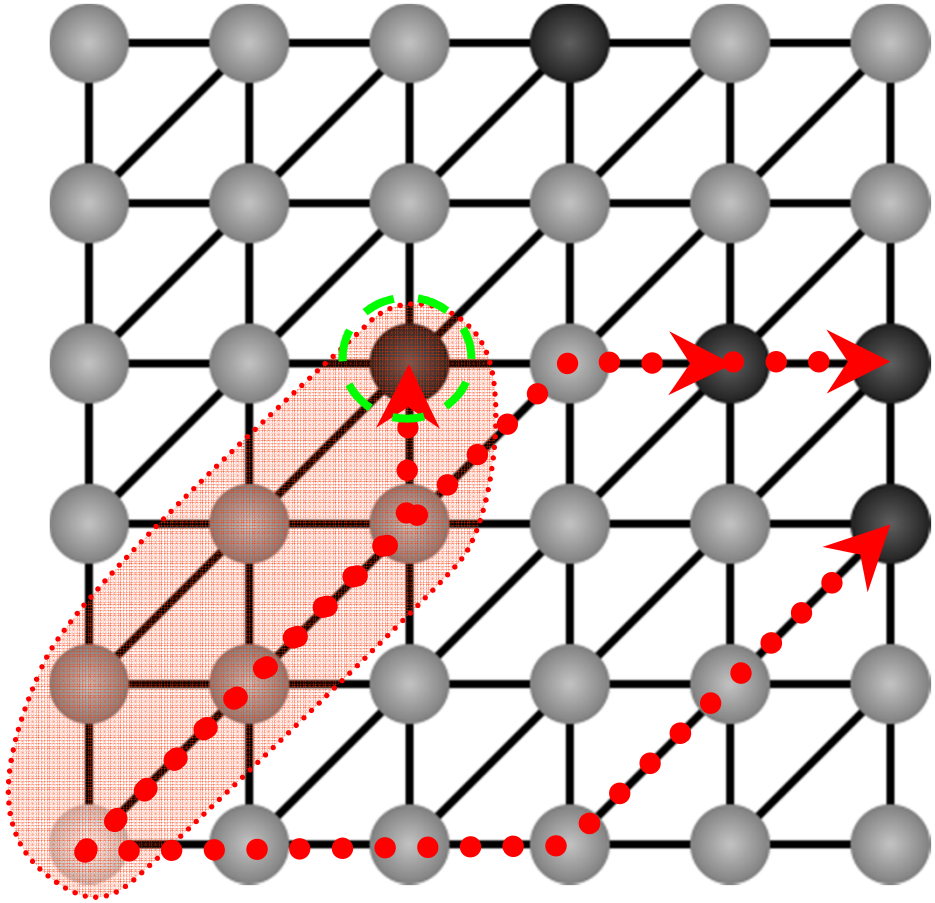
Enhanced Shortest Path Routing



MANCHESTER 1824

The University of Manchester

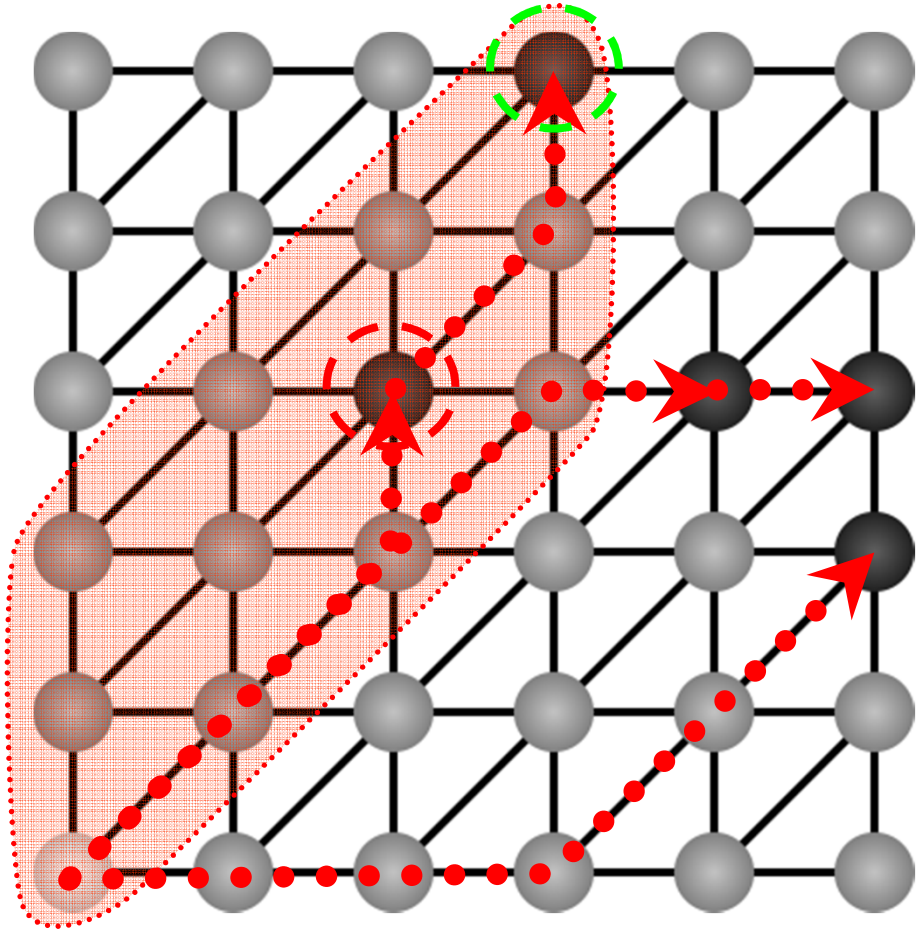
Enhanced Shortest Path Routing



MANCHESTER 1824

The University of Manchester

Enhanced Shortest Path Routing

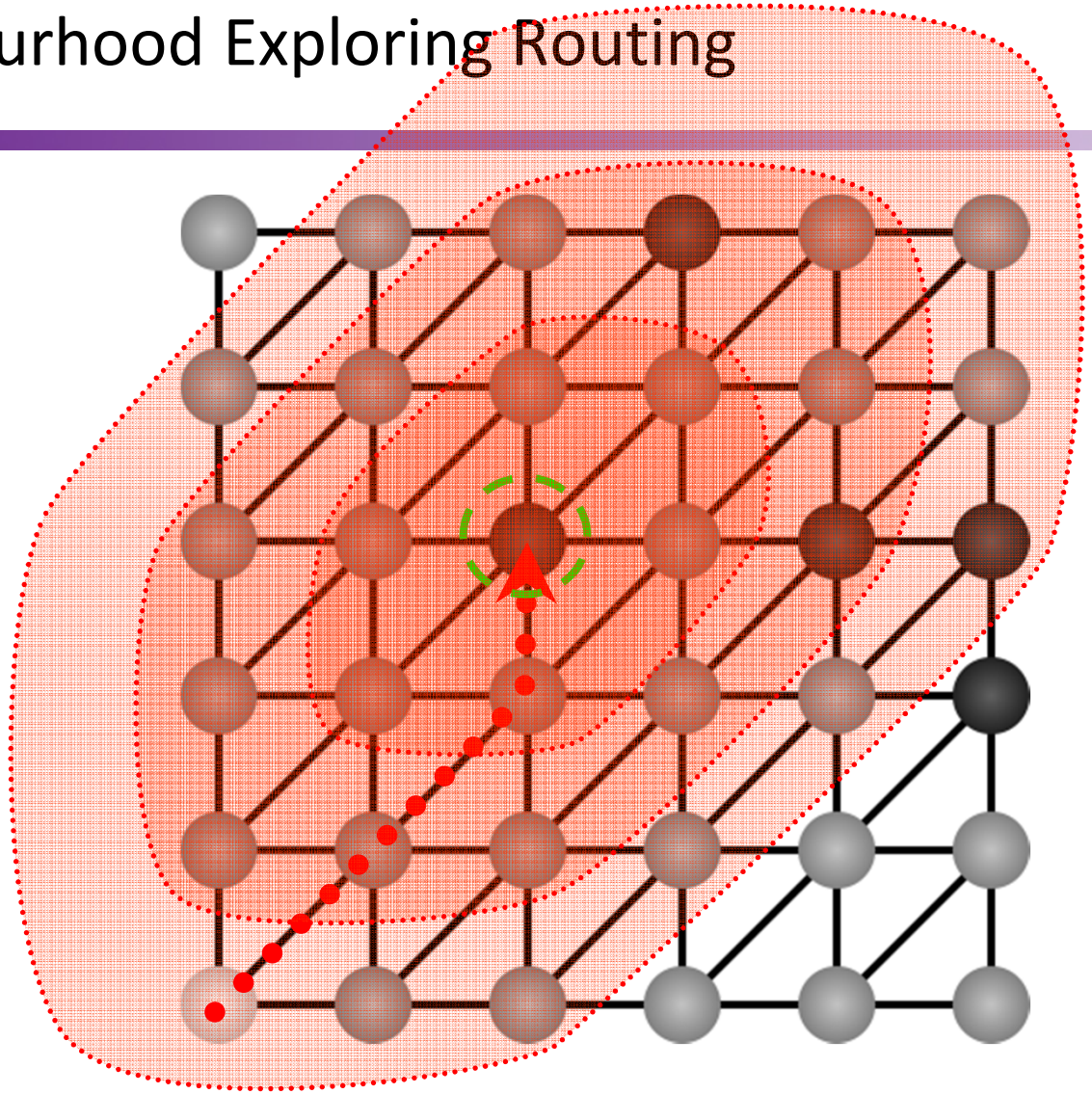


Each node looks for the best connection in the shortest path

MANCHESTER 1824

The University of Manchester

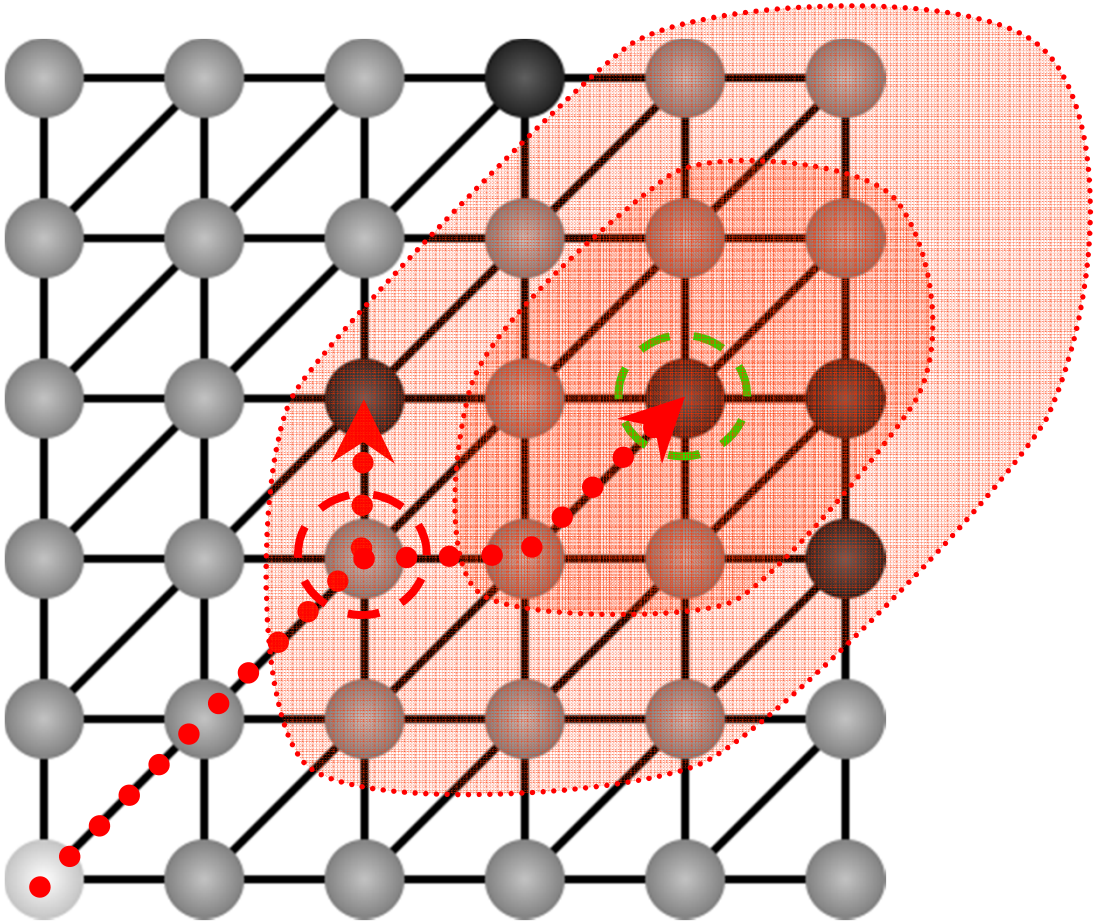
Neighbourhood Exploring Routing



MANCHESTER 1824

The University of Manchester

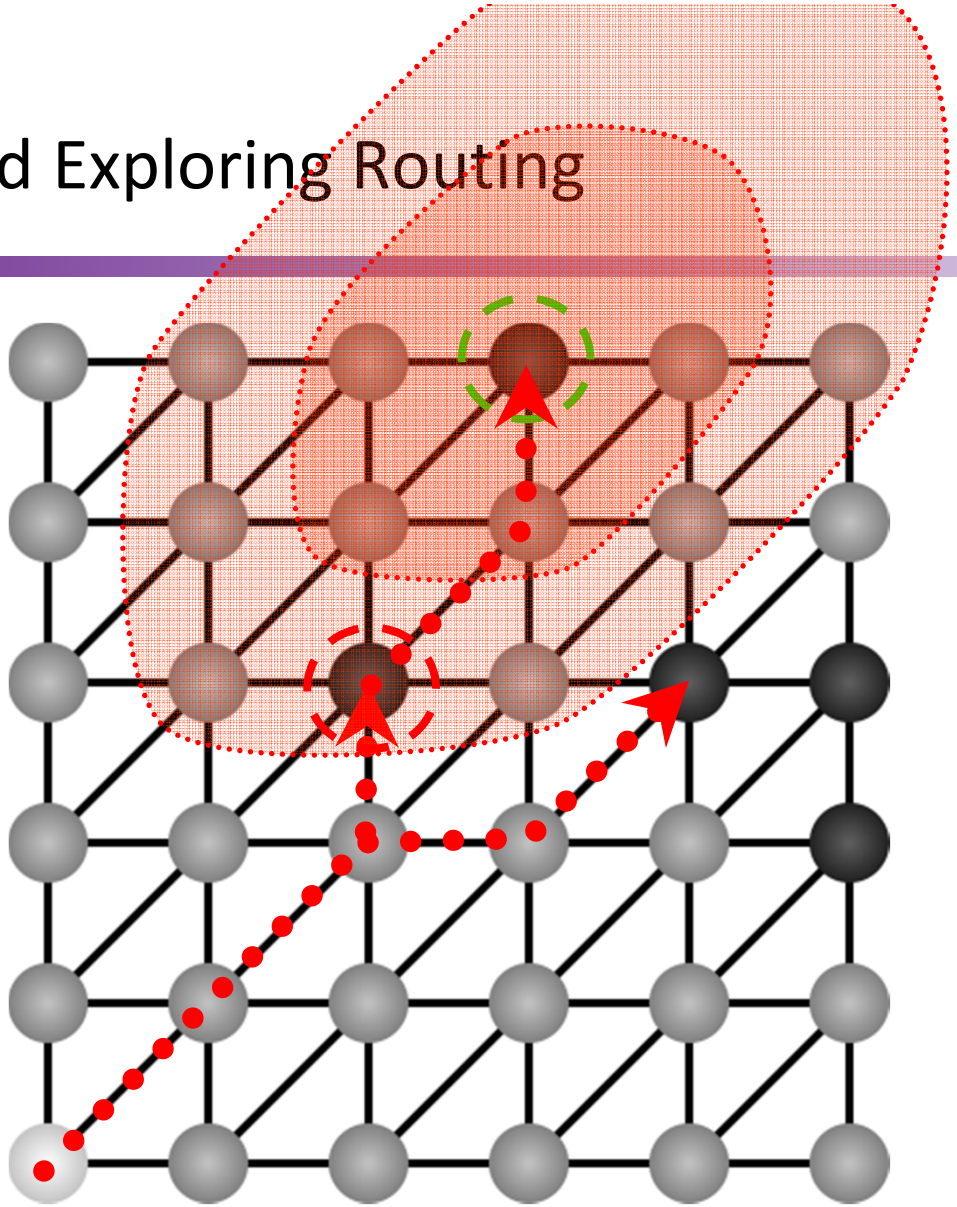
Neighbourhood Exploring Routing



MANCHESTER 1824

The University of Manchester

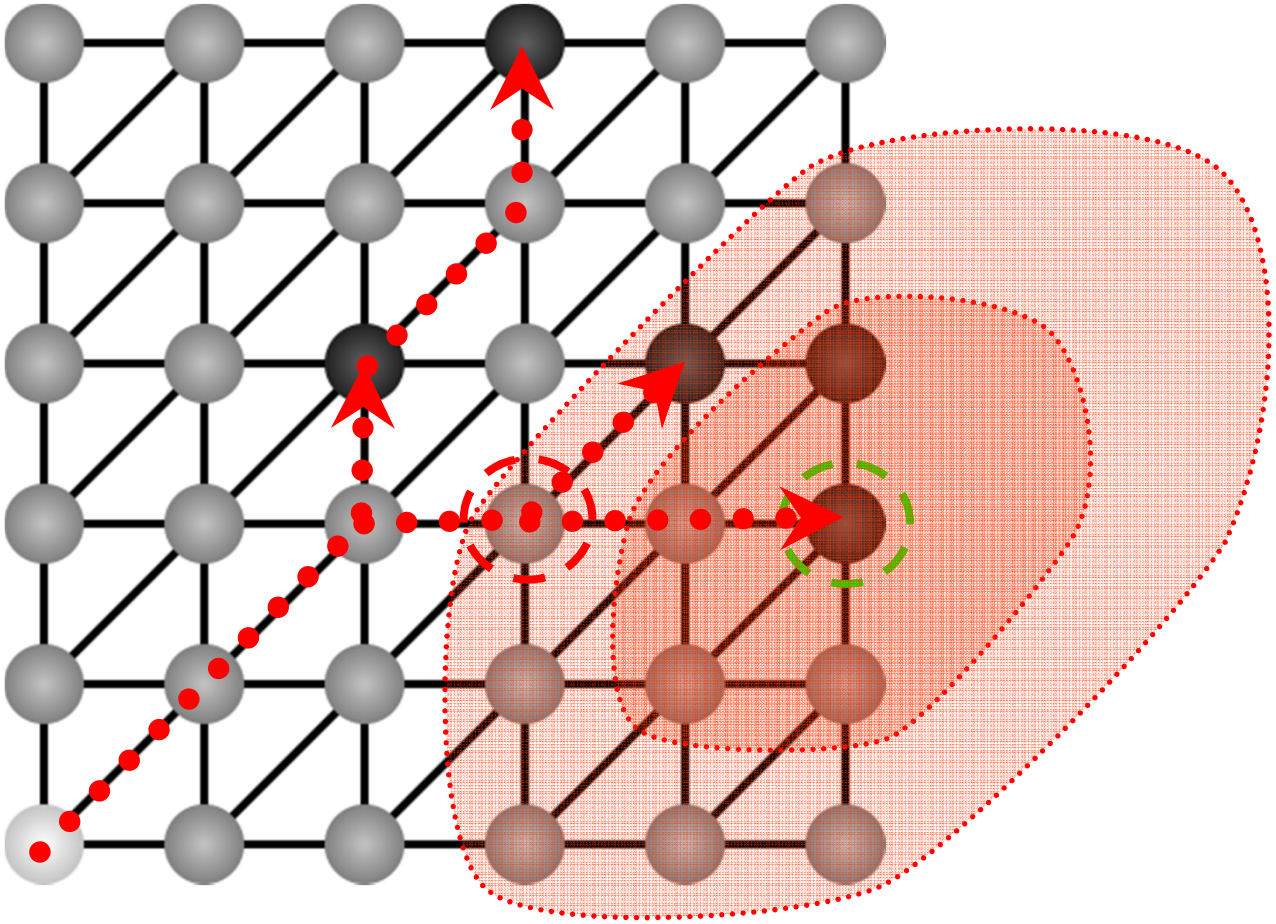
Neighbourhood Exploring Routing



MANCHESTER 1824

The University of Manchester

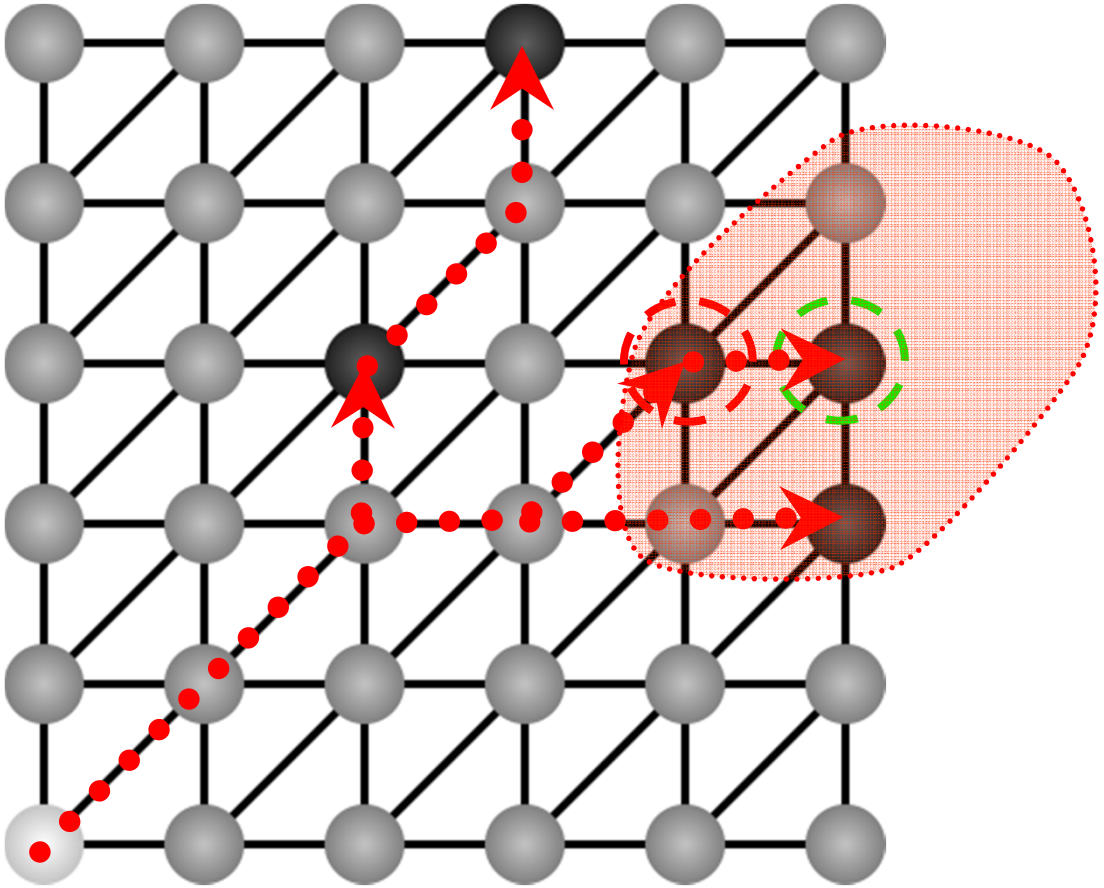
Neighbourhood Exploring Routing



MANCHESTER 1824

The University of Manchester

Neighbourhood Exploring Routing



Each node looks for the best connection around.

The University of Manchester

MANCHESTER 1824

SpiNNaker
a universal Spiking Neural Network architecture

Experimental Set-up

MANCHESTER 1824

The University of Manchester

SpiNNaker Architecture

- Essential info to motivate the performed analyses
  - Multicast routes are constructed before starting the neural application
    - Slower generation means larger waiting time before execution.

    Analyse route generation times
  - Limited Link Bandwidth
    - ~250 MB/s, need to reduce used bandwidth

    Assess bandwidth utilization and load balancing
  - Reduced number of routing table entries
    - 1024 masked entries per router, cannot exceed this number

    Measure routing table utilization
  - 18 cores per chip, of which 16 are typically used for the application
    - ~1000 neurons can be simulated per core
    - We need to avoid the network becoming the bottleneck

    Calculate number of supported neurons per core

How can we add fault tolerance?

Work in Progress

MANCHESTER 1824

The University of Manchester

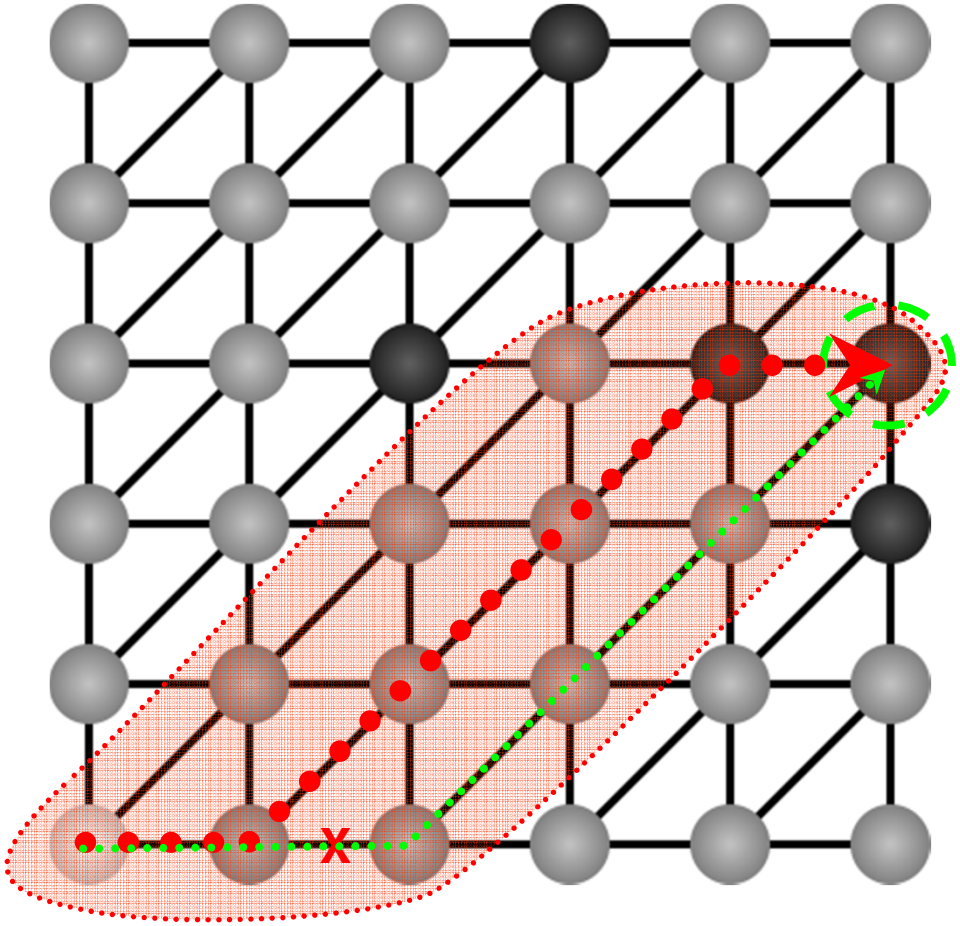
FT techniques

- Want to avoid backtracking to reduce memory footprint
  - Greedy techniques implemented (under study yet)
    - Build the route one step at a time and react to failures locally
    - Based on the previously discussed algorithms
- Need to exploit the rich connectivity of the topology
  - …but need to ensure livelock free
    - Hence we should restrict to somehow limited routes
    - Mostly shortest routes
- Previous performance metrics still apply
  - Fast route generation
  - Reduced bandwidth utilization
  - Reduced routing table utilization

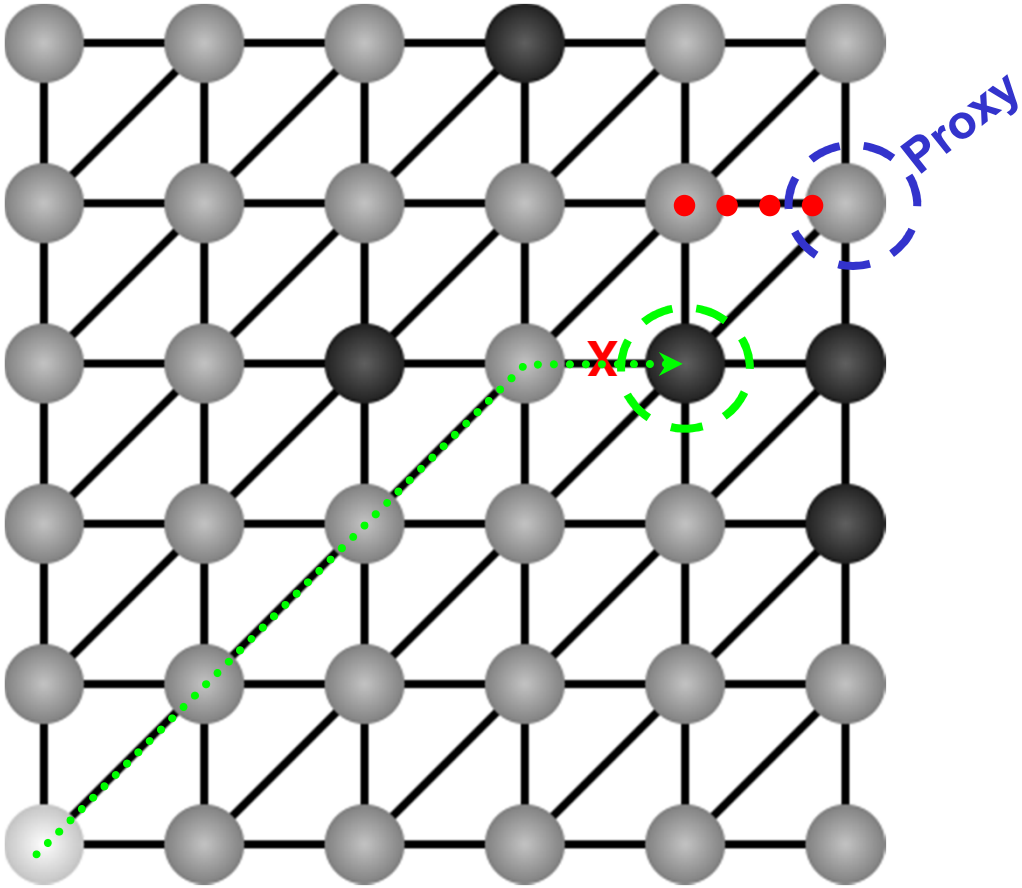
MANCHESTER 1824

The University of Manchester

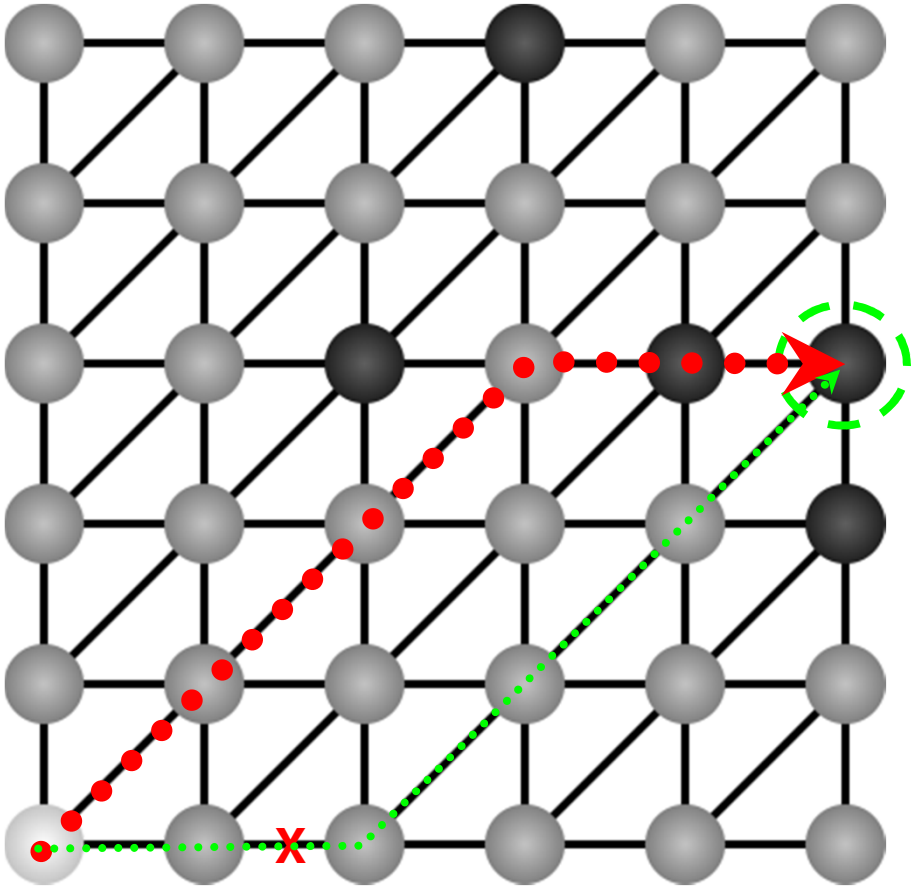
Diverting within the *minimal path* area



Look for a Proxy (*minimal path* not necessary)



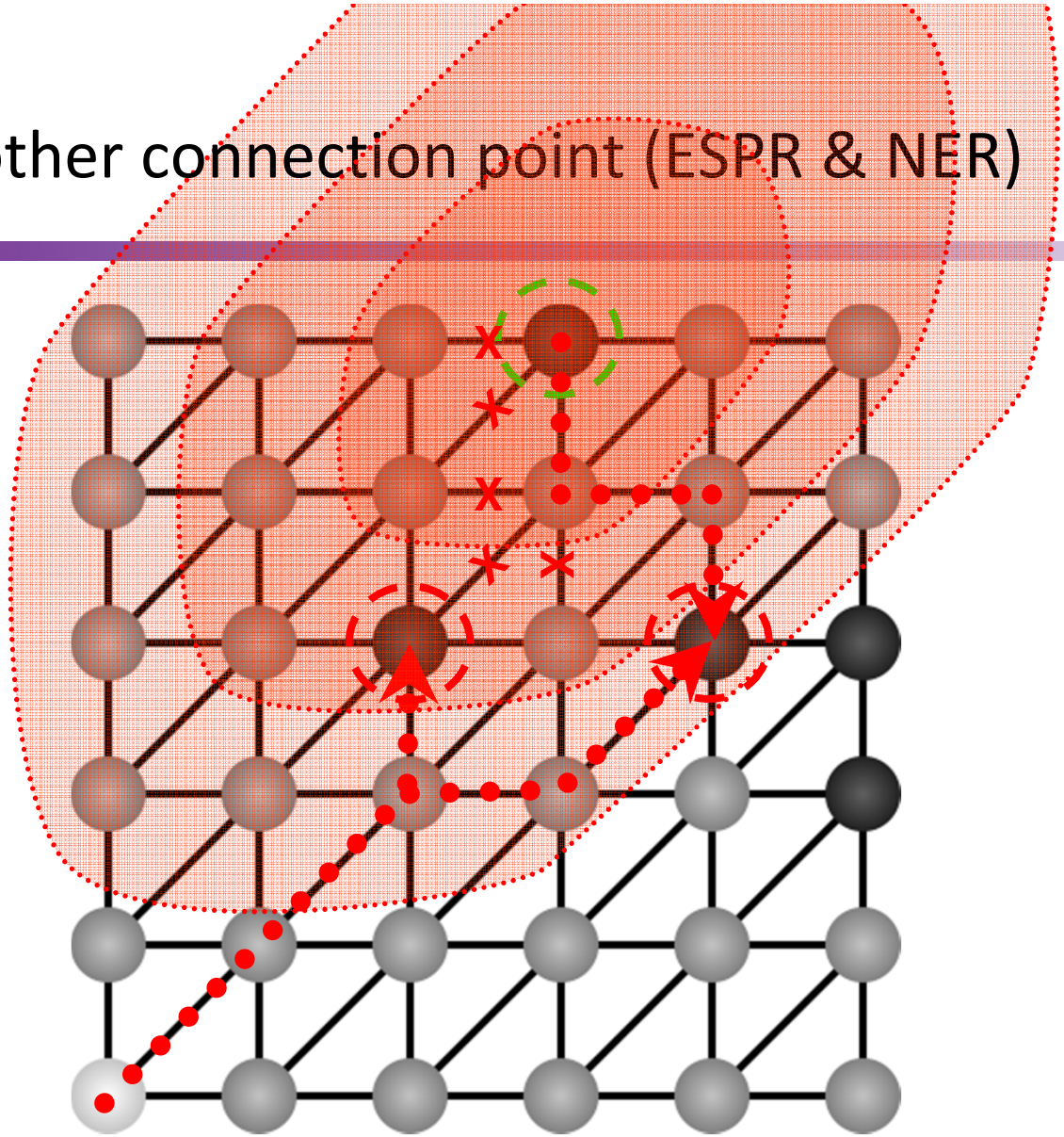
Try other order of dimensions



Cannot continue in this route

- **No Possible Diversion**
- **No Possible Proxy**

Look for another connection point (ESPR & NER)



Reaching the closest connection point is not possible

- **Paths in both dimension are blocked**
- **No Possible Diversion**
- **No Possible Proxies**