# Fetch Data Transfer Protocol


THE UNIVERSITY of MISSISSIPPI

Clay McLeod

University of Mississippi

November 15, 2015

# Sources

- Presentation available at `http://bit.ly/wsn-final`.
- Based on ideas in Werner-Allen, Geoffrey, Konrad Lorincz, Mario Ruiz, Omar Marcillo, Jeff Johnson, Jonathan Lees, and Matt Welsh. 2006. Deploying a Wireless Sensor Network on an Active Volcano. Internet Computing, IEEE 10 (2). IEEE: 1825.
  - Full text is viewable at `http://www.cs.harvard.edu/~mdw/papers/volcano-ieeeic06.pdf`
- Proposal can be found at `http://claymcleod.github.io/papers/engr691-final-project/paper.pdf`

# Motivation

- Paper discusses authors implementing a wireless sensor network (WSN) to track volcanic activity remotely.

# Motivation

- Paper discusses authors implementing a wireless sensor network (WSN) to track volcanic activity remotely.
- What problems did they encounter?

# Motivation

- Paper discusses authors implementing a wireless sensor network (WSN) to track volcanic activity remotely.
- What problems did they encounter?
  - Gathering accurate data from events
  - High availability of sensors
  - Time synchronization across nodes
  - Low radio bandwidth to work with
  - Node failure could impact communication
  - Radio links are lossy and frequently asymmetrical

# Motivation

- Paper discusses authors implementing a wireless sensor network (WSN) to track volcanic activity remotely.
- What problems did they encounter?
  - Gathering accurate data from events
  - High availability of sensors
  - Time synchronization across nodes
  - Low radio bandwidth to work with
  - Node failure could impact communication
  - Radio links are lossy and frequently asymmetrical
- **Summary:** the solution in my final project aims to alleviate any subproblem related to data collectionand time synchronization, usually by abstracting them away.

# Motivation

- Paper discusses authors implementing a wireless sensor network (WSN) to track volcanic activity remotely.
- What problems did they encounter?
  - Gathering accurate data from events X
  - High availability of sensors X
  - Time synchronization across nodes ✓
  - Low radio bandwidth to work with ✓
  - Node failure could impact communication ✓
  - Radio links are lossy and frequently asymmetrical ✓
- **Summary:** the solution in my final project aims to alleviate any subproblem related to data collectionand time synchronization, usually by abstracting them away.

# Fetch Data Collection

- **Goal**: build a reliable data-collection protocol that is used to retrieve buffered data from each node over a multihop network.

# Fetch Data Collection

- **Goal**: build a reliable data-collection protocol that is used to retrieve buffered data from each node over a multihop network.
- **Idea**

# Fetch Data Collection

- **Goal**: build a reliable data-collection protocol that is used to retrieve buffered data from each node over a multihop network.
- **Idea**
  1. The sensor node breaks its data down into 256 bytes, then tags these blocks with timestamps and sequence numbers.

# Fetch Data Collection

- **Goal**: build a reliable data-collection protocol that is used to retrieve buffered data from each node over a multihop network.
- **Idea**
  1. The sensor node breaks its data down into 256 bytes, then tags these blocks with timestamps and sequence numbers.
  2. The laptop then sends packets out to the target node ID identifying which sequence numbers it is missing from that node.

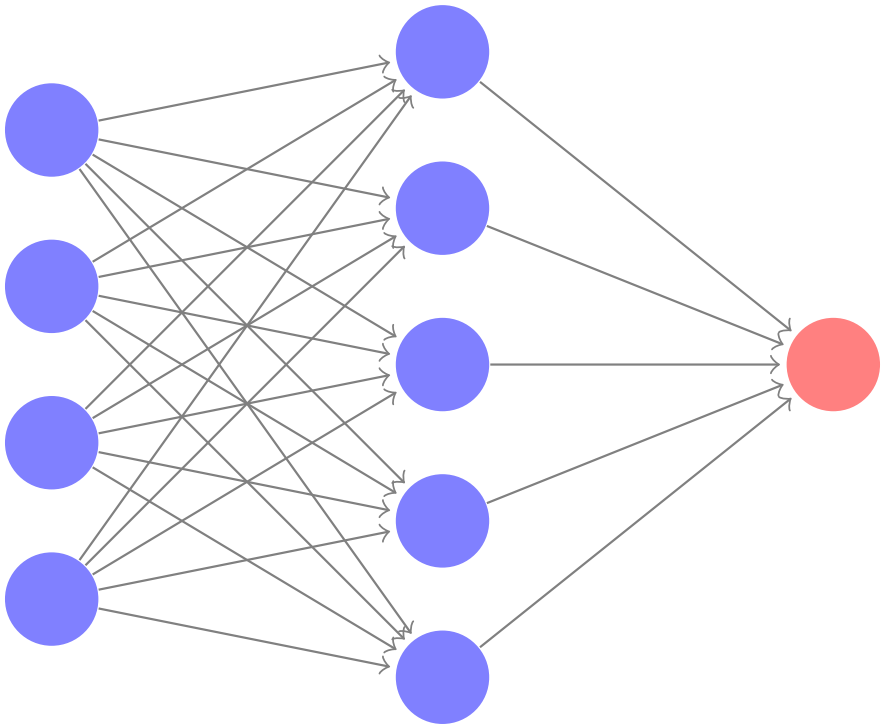
# Fetch Data Collection

- **Goal**: build a reliable data-collection protocol that is used to retrieve buffered data from each node over a multihop network.
- **Idea**
  1. The sensor node breaks its data down into 256 bytes, then tags these blocks with timestamps and sequence numbers.
  2. The laptop then sends packets out to the target node ID identifying which sequence numbers it is missing from that node.
  3. In turn, the node will send the missing chunks until the laptop indicates it has received all sequences.

# Fetch Data Collection

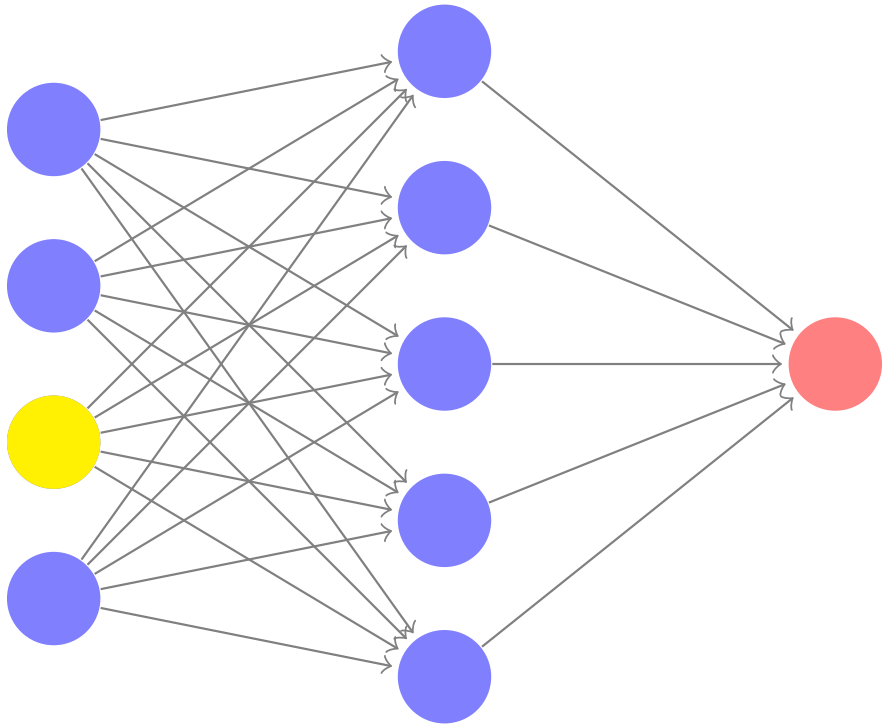
- **Goal**: build a reliable data-collection protocol that is used to retrieve buffered data from each node over a multihop network.
- **Idea**
  1. The sensor node breaks its data down into 256 bytes, then tags these blocks with timestamps and sequence numbers.
  2. The laptop then sends packets out to the target node ID identifying which sequence numbers it is missing from that node.
  3. In turn, the node will send the missing chunks until the laptop indicates it has received all sequences.
  4. Because the network is sparse, the laptop uses **flooding** to request data from the network.

# Visualization



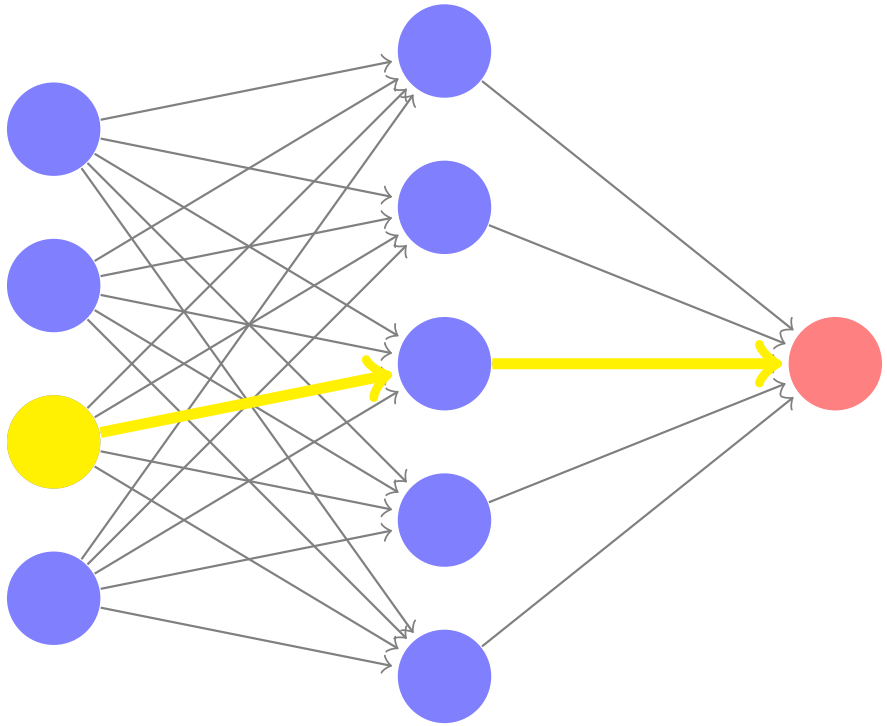
# Visualization

A node indicates that an activity has occurred based on some input threshold.



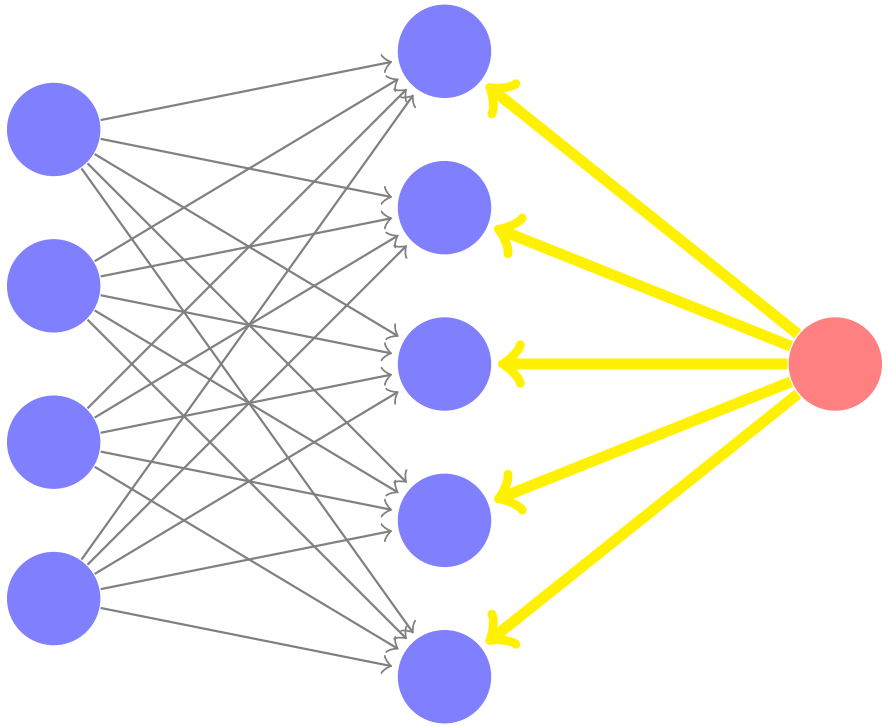
# Visualization

That node communicates back to the base station, letting it know that an event has been perceived



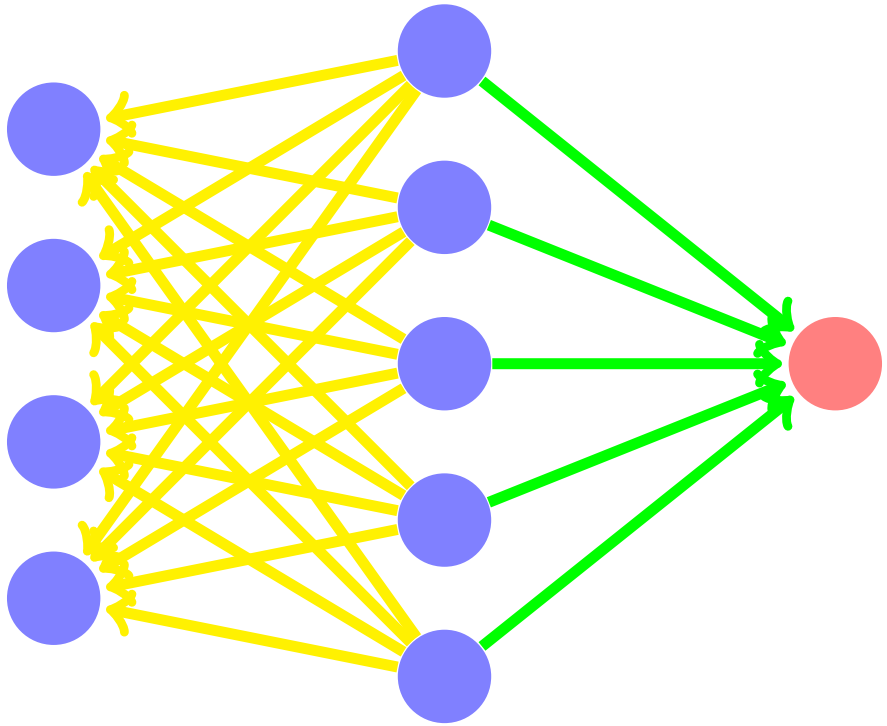
# Visualization

When the base station receives enough signal, a data collection is initiated



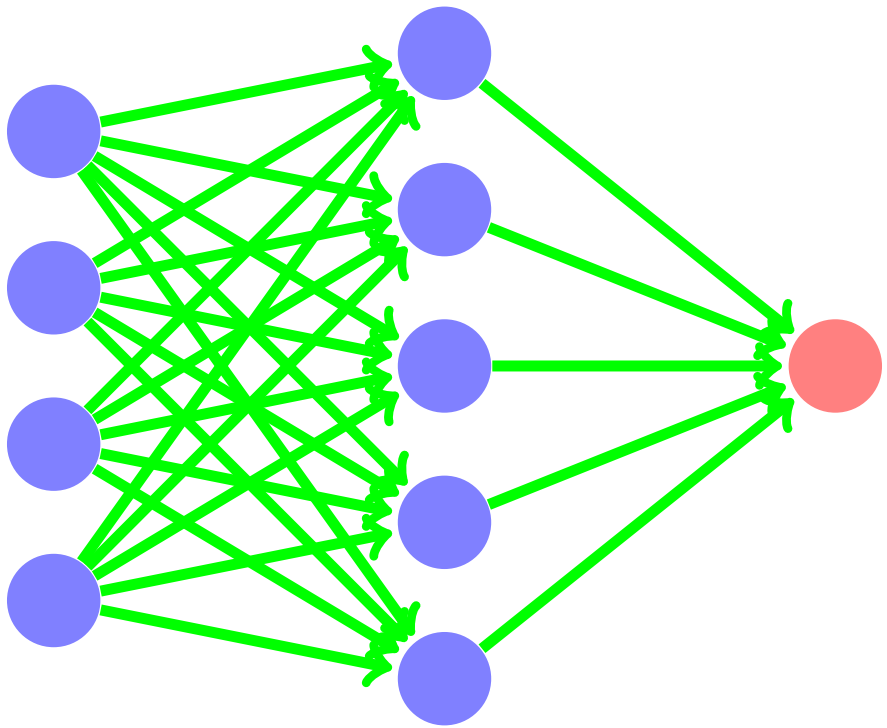
# Visualization

Nodes closest to the base station attempt to transfer their data to the base station while the request propigates through the WSN



# Visualization

Eventually, all nodes are communicating with the base station



# Experiment

- Designate one node as a **master** node (connected through serial connection to laptop) and two nodes as **slave** nodes.
  - *Due to the limited number of supplies, only 2 slave nodes can be fabricated*

# Experiment

- Designate one node as a **master** node (connected through serial connection to laptop) and two nodes as **slave** nodes.
  - *Due to the limited number of supplies, only 2 slave nodes can be fabricated*
- Using a light sensor attached to the circuit board, we can simluate our own event by setting a threshold by how much light is present to trigger an event.

# Experiment

- Designate one node as a **master** node (connected through serial connection to laptop) and two nodes as **slave** nodes.
  - *Due to the limited number of supplies, only 2 slave nodes can be fabricated*
- Using a light sensor attached to the circuit board, we can simluate our own event by setting a threshold by how much light is present to trigger an event.
- Data about the amount of light present will be constantly recorded and trasmitted to the base station once an event has occurred using the protocol described earlier.

# Comparison with Paper

| Description | Paper | Project | Comparable |
|---|---|---|---|
| Data Type | Seismic Activity | Light Presence | X |
| Data Frequency | Continuous | Continuous | ✓ |
| Node Distance | 200-400m | $< 10$m | X |
| Network | Lossy | Consistent | X |
| Hardware | IEEE 802.15.4 | IEEE 802.15.4 | ✓ |
| Software | TinyOS | TinyOS | ✓ |

# Block Storage

- Configuration File
  - Declare `BlockStorageC` and hook up to implementation

# Block Storage

- Configuration File
  - Declare `BlockStorageC` and hook up to implementation
- Main File
  - Explicitly state that the implementation uses `interface BlockRead` and `interface BlockWrite`
  - Implement the following methods on the interface
    - `BlockWrite.writeDone`
    - `BlockWrite.eraseDone`
    - `BlockWrite.syncDone`
    - `BlockRead.readDone`
    - `BlockRead.computeCrcDone`

# Block Storage

- Configuration File
  - Declare `BlockStorageC` and hook up to implementation
- Main File
  - Explicitly state that the implementation uses `interface BlockRead` and `interface BlockWrite`
  - Implement the following methods on the interface
    - `BlockWrite.writeDone`
    - `BlockWrite.eraseDone`
    - `BlockWrite.syncDone`
    - `BlockRead.readDone`
    - `BlockRead.computeCrcDone`
- Potentially define interface for storing data

# Results

# Results

*In Progress (TBD)*

# Questions?