BoilerNet
Final Design Review

Akshath Raghav Ravikiran, Aneesh Poddutur,
Gautam Nambiar, Gokul Harikrishnan

04/30/2025

PURDUE UNIVERSITY®

Team Introduction

Gautam Nambiar

BSCmpE est. May 2025

Akshath Raghav Ravikiran
BSCmpE est. May 2025

Gokul Harikrishnan

BSCmpE est. May 2025

Aneesh Poddutur

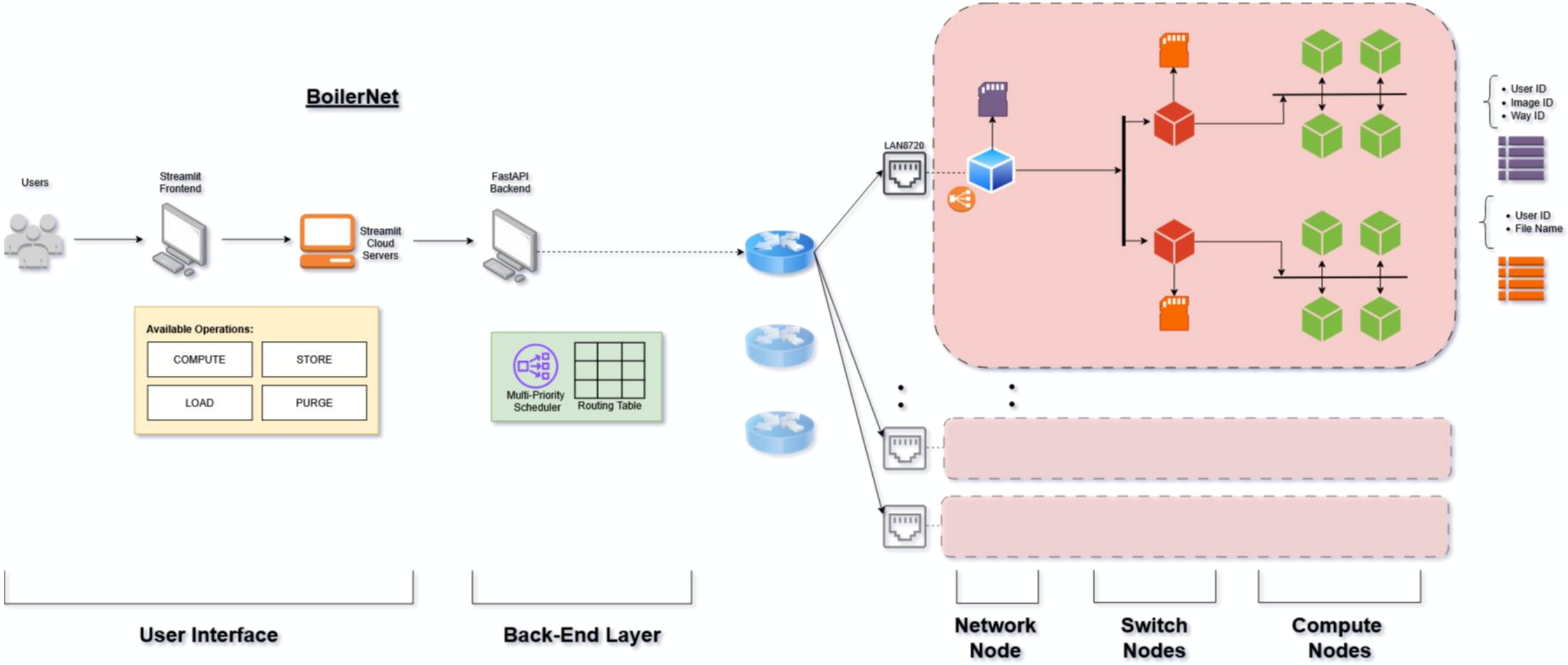
BSEE est. May 2025

PURDUE UNIVERSITY

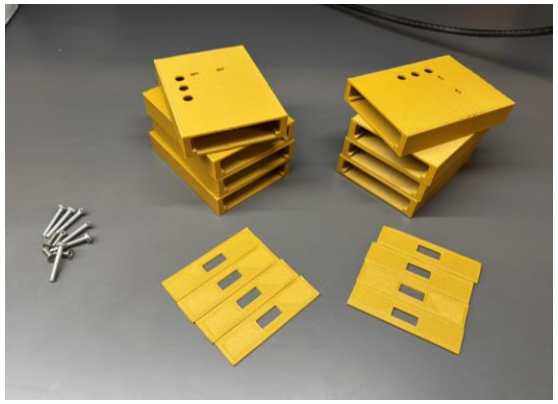
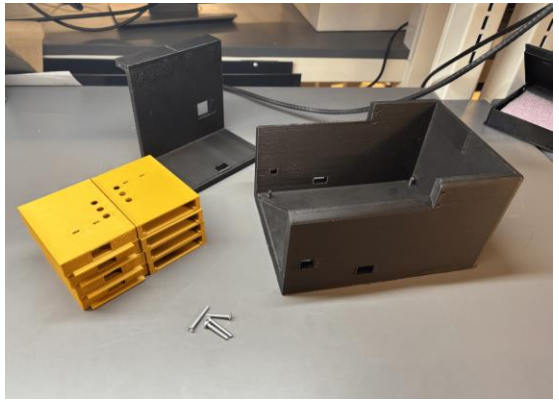
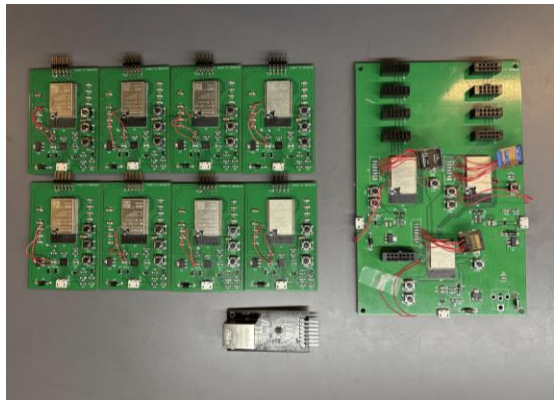
BoilerNet

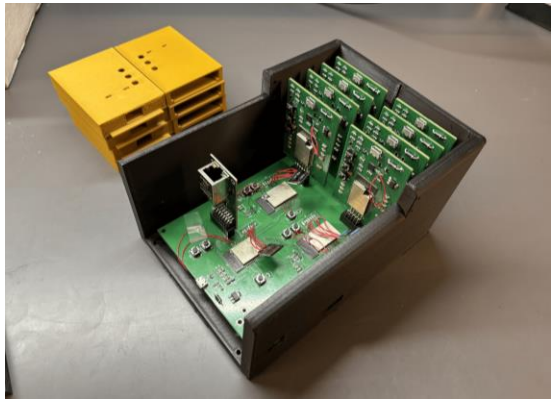
- Portable
- Swappable Compute Workloads
- Low Power
- Scalable
- Throughput-optimized

Block Diagram



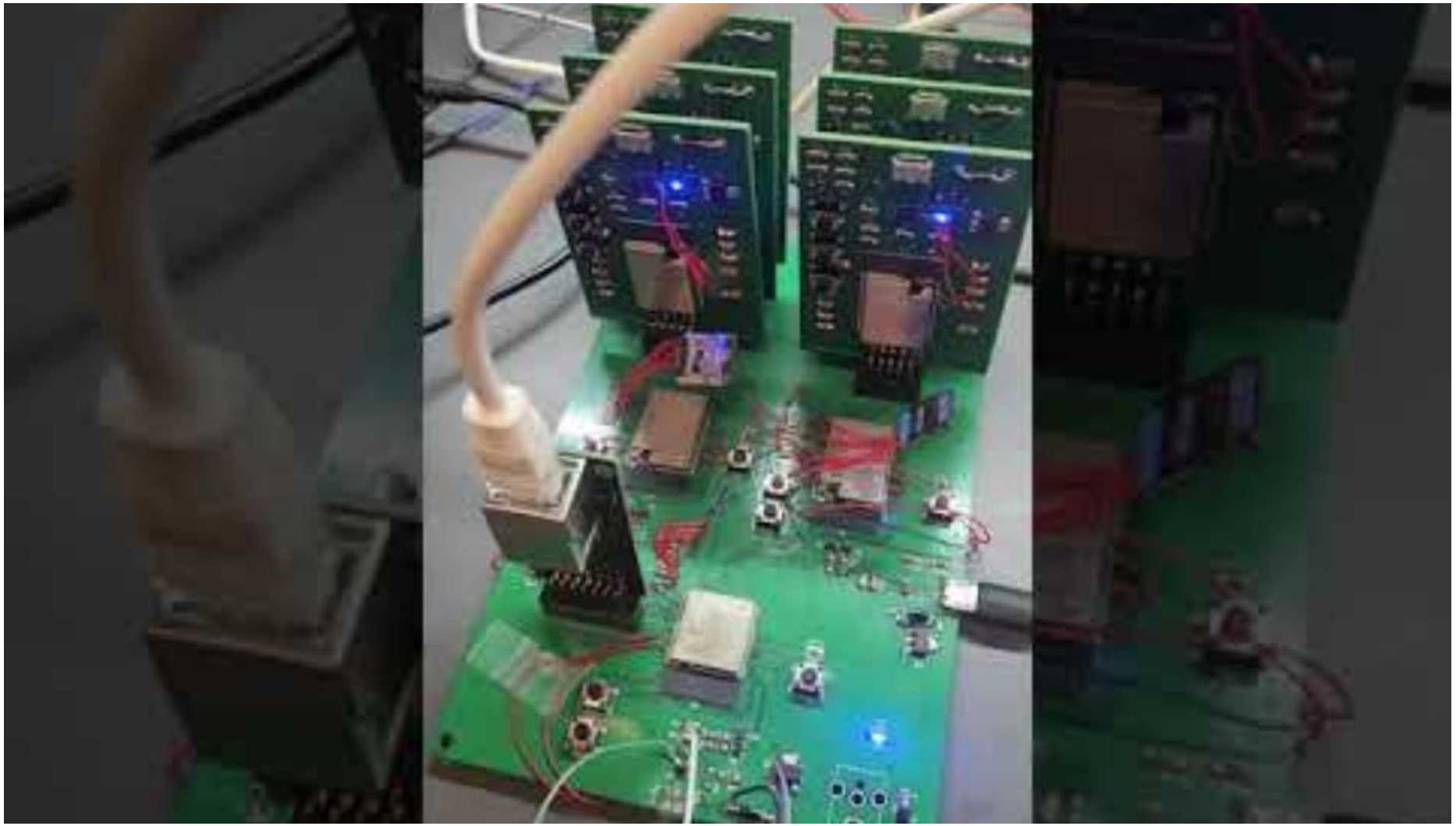
Mechanical Design



PURDUE UNIVERSITY

System Integration



PURDUE UNIVERSITY

System Requirements & Status

- Requirement 1: The User-Interface will allow the User to choose the Operation and define a Job with a (limited) number of Tasks for the cluster to work on. This improves User Experience.

- Requirement 2: The Backend will be responsible for buffering Jobs and sending in Tasks serially into the Network Node. It will contain a multi-priority-messaging queue. This ensures no Operation request is starved, and helps in prioritizing I/O heavy operations.

- Requirement 3: The Network Node will send the entire task into one of the Ways. Way Selection must happen inside the Network Node. This promotes scalability by decoupling responsibilities.

- Requirement 4: The Internal Switch Nodes are responsible for Data Parallelism at the Compute level, and must deal with Compute failures dynamically.

System Requirements & Status (Contd.)

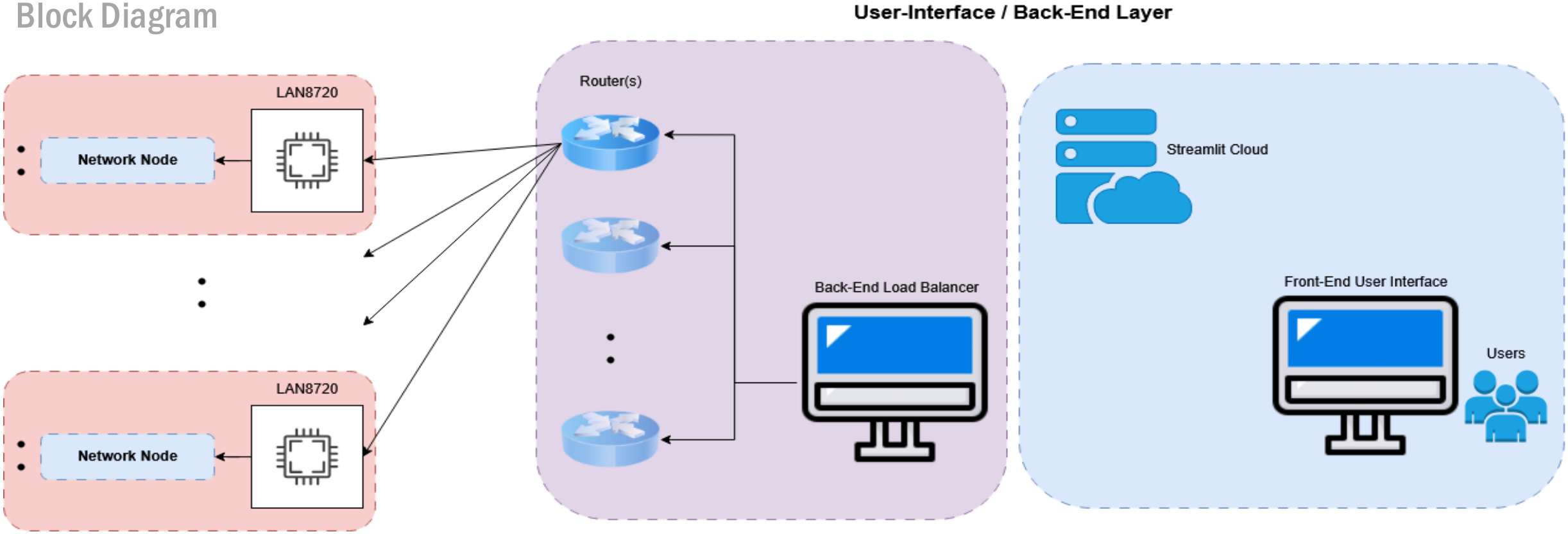
- Requirement 6: The Compute Nodes will buffer the PNG-encoded packets as they arrive, and will perform decoding in-memory at runtime. This ensures parallelism can be abstracted away to switches.

- Requirement 7: The Compute Nodes are to be developed onto Cartidge-like PCBs, allowing for slotting and unslotting with ease. This allows for swappable workloads.

- Requirement 8: The Compute Cluster, as a whole, must be contained within it's 3-D printed enclosure, made in Black and Gold colors. This ensures portability.

User-Interface & Back-End Layer

Block Diagram



User-Interface Highlights:

- Supports multi-file upload per Job with task-level operation selection
- Polls Backend for task status and auto-refreshes UI on completion
- Enables repeated STORE, COMPUTE, LOAD, PURGE on any Task

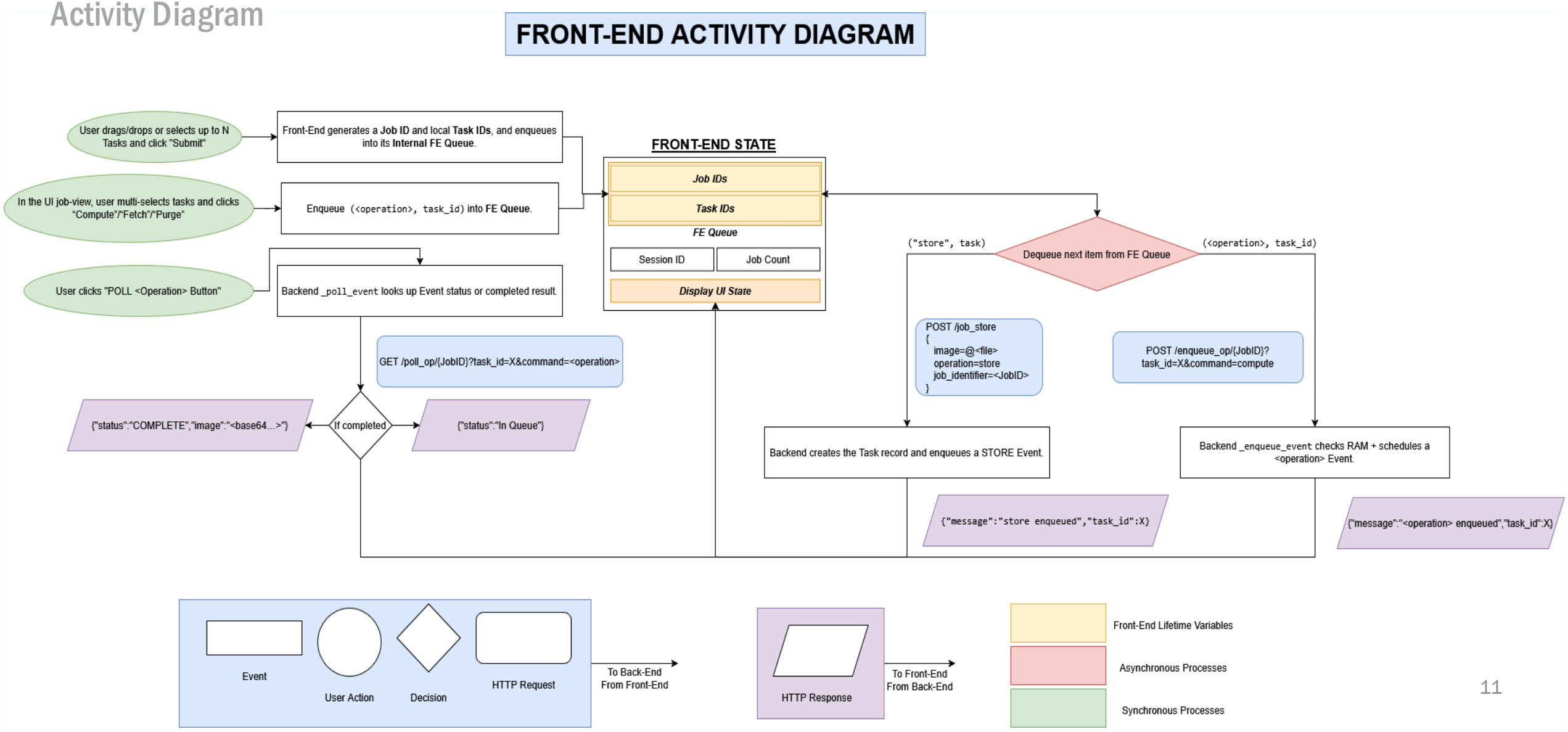
Back-End Highlights:

- Built with FastAPI and AsyncIO for non-blocking operation handling
- Uses a multi-priority scheduler with RAM-aware enqueue policies
- Dedicated worker loop ensures COMPUTE and non-COMPUTE tasks are multiplexed

PURDUE UNIVERSITY

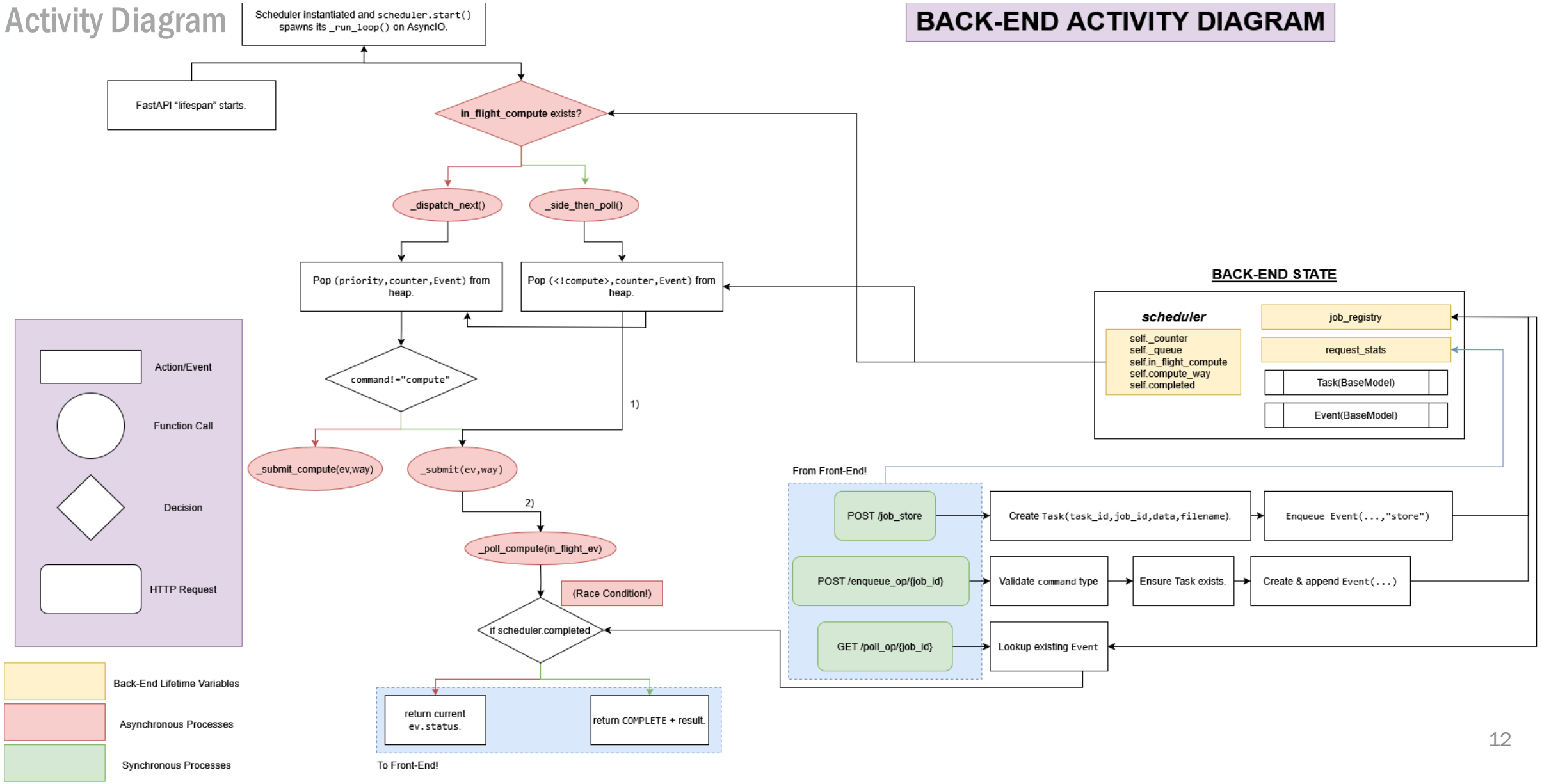
User-Interface & Back-End Layer

Activity Diagram



User-Interface & Back-End Layer

Activity Diagram



User-Interface & Back-End Layer

Specifications

Specification Relevance

- The Back-End must be able to guarantee response to any User-Interface request in under 50ms.
- The Back-End must prioritize STORE operations, and enable tracking & polling of in-flight COMPUTE operations.
- The Back-End must cancel transactions if total RAM usage exceeds 30\% of available RAM
- The Back-End must be able to complete a 2KB transfer into the Network Node in under 100ms.

Specification Testability

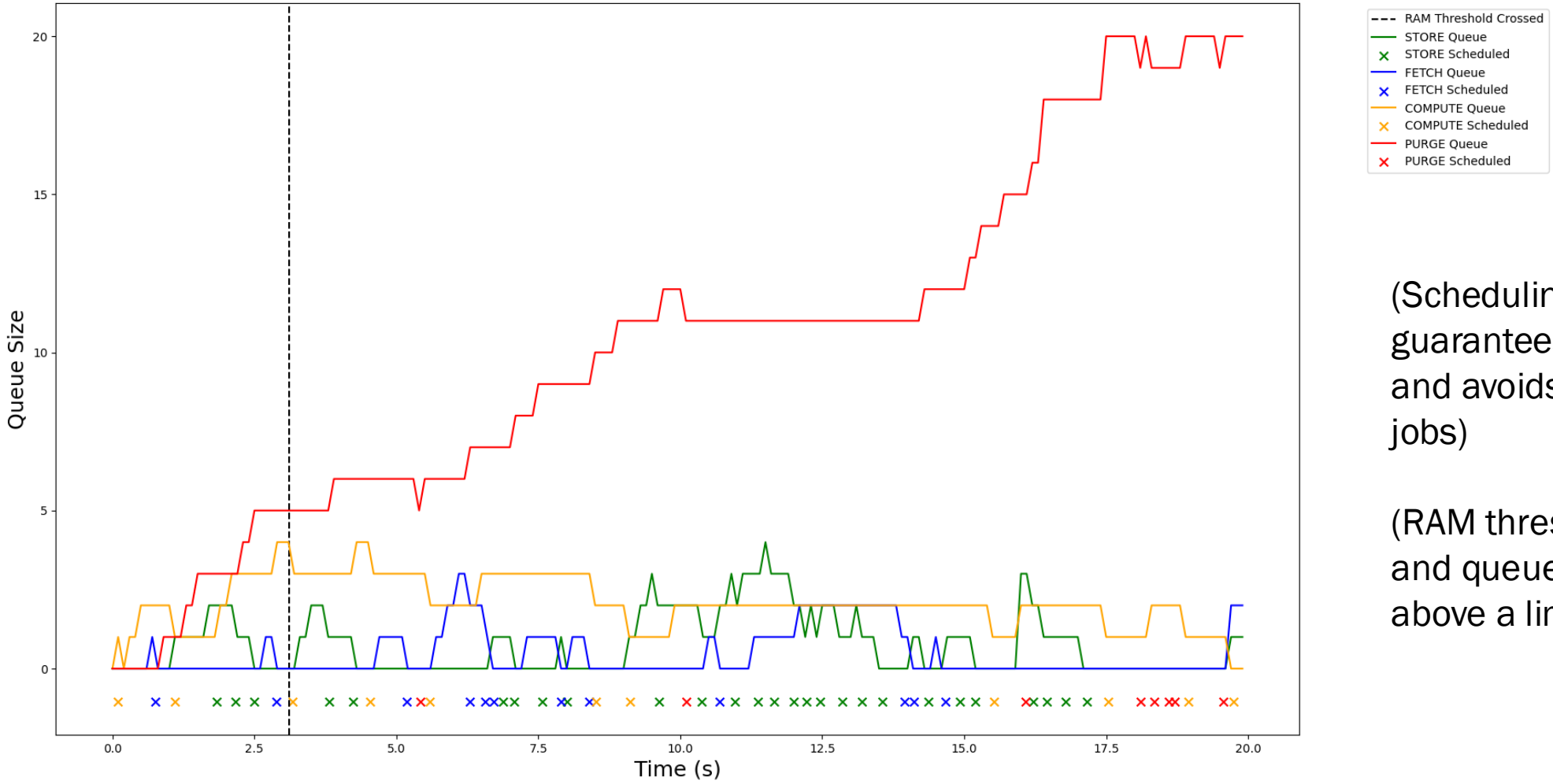
- Have a `app.middleware` endpoint, tracking lifetime of servicing all endpoints. Average out RTT throughout all requests.
- Stress Test the Scheduler layer; simulate randomized requests, delays and failures.
- Track RAM Usage in above Stress Test, and visualize peak queue-sizes.
- Timing Summaries across I/O heavy operations -- STORE and FETCH.

Specification Motivation

- Ensures that the user interface remains responsive and avoids timeouts.
- Even under high system load, the scheduler ensures that STORE operations are not starved by longer-running commands. COMPUTE operations will be polled between trivial-ops.
- Prevents device failure!
- Guarantees minimal latency when synchronizing job/task updates between Backend and Nodes.

User-Interface & Back-End Layer

Specification Evidence



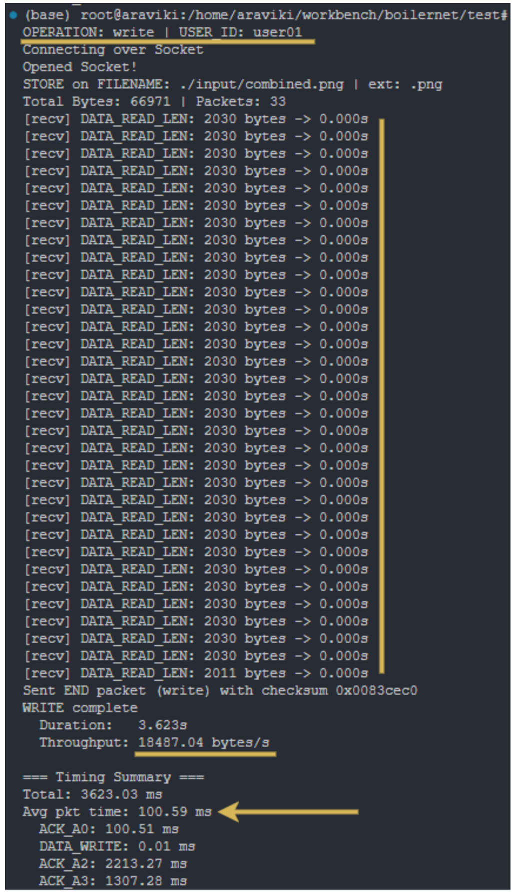
(Scheduling logic guarantees STORE priority, and avoids starvation of jobs)

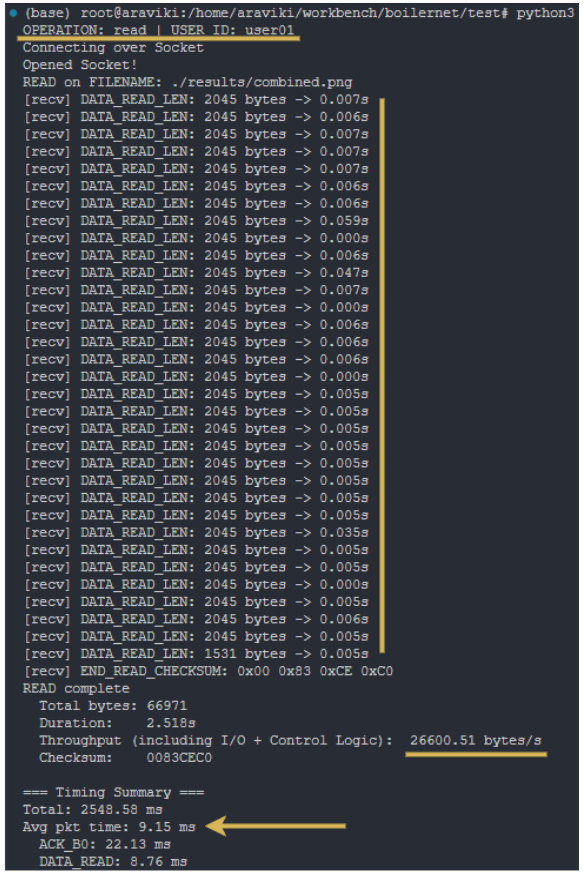
(RAM threshold tracked, and queues don't grow above a limit)

https://github.com/AkshathRaghav/boilerNAS/blob/main/test/test_schedule.py

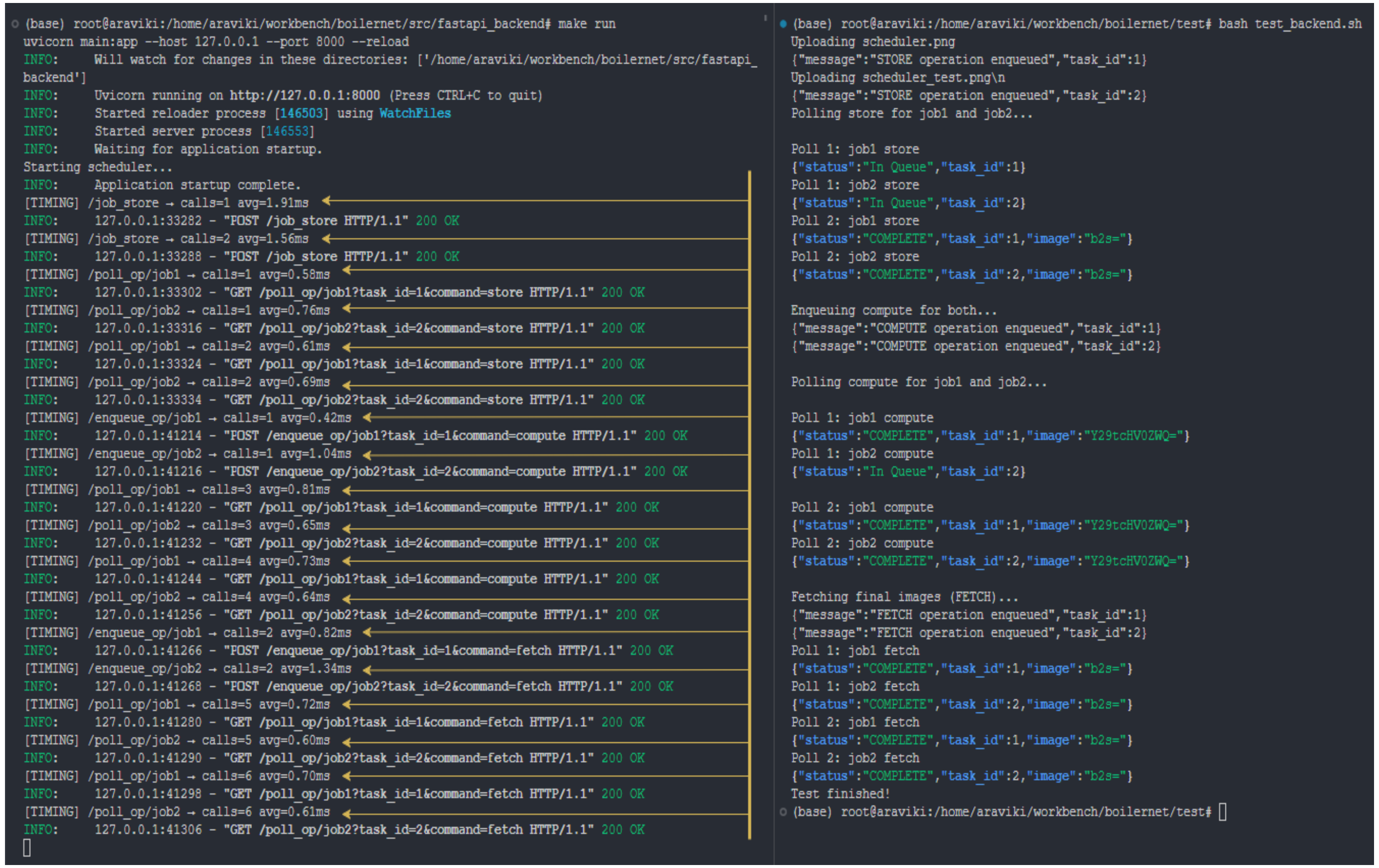
User-Interface & Back-End Layer

Specification Evidence





Back-End -> Network-Node 2KB Transfer < 100ms.



Back-End -> User-Interface Critical Path < 50ms.

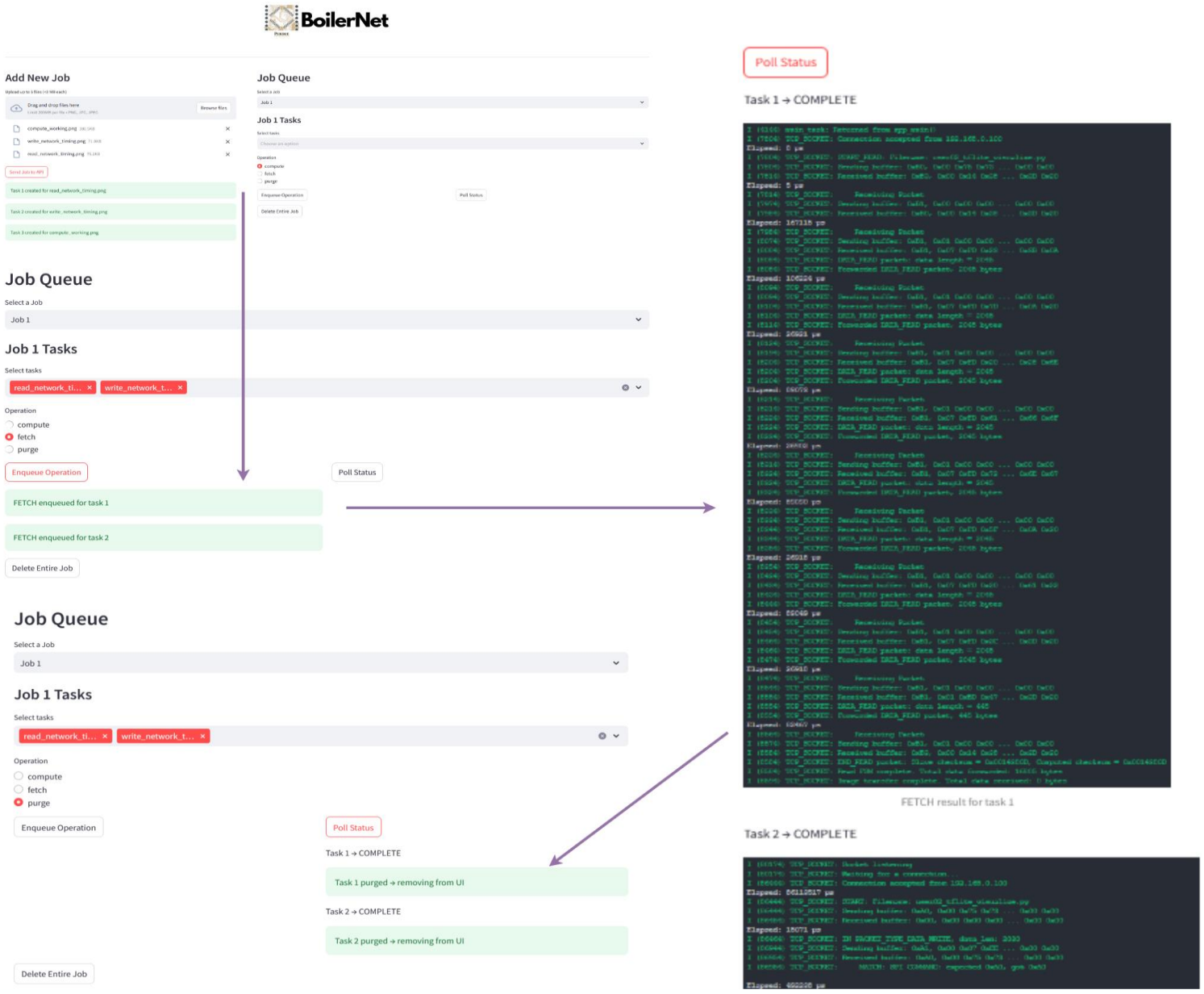
PURDUE UNIVERSITY.

https://github.com/AkshathRaghav/boilerNAS/blob/main/test/test_backend.sh

https://github.com/AkshathRaghav/boilerNAS/blob/main/test/test_routines.py

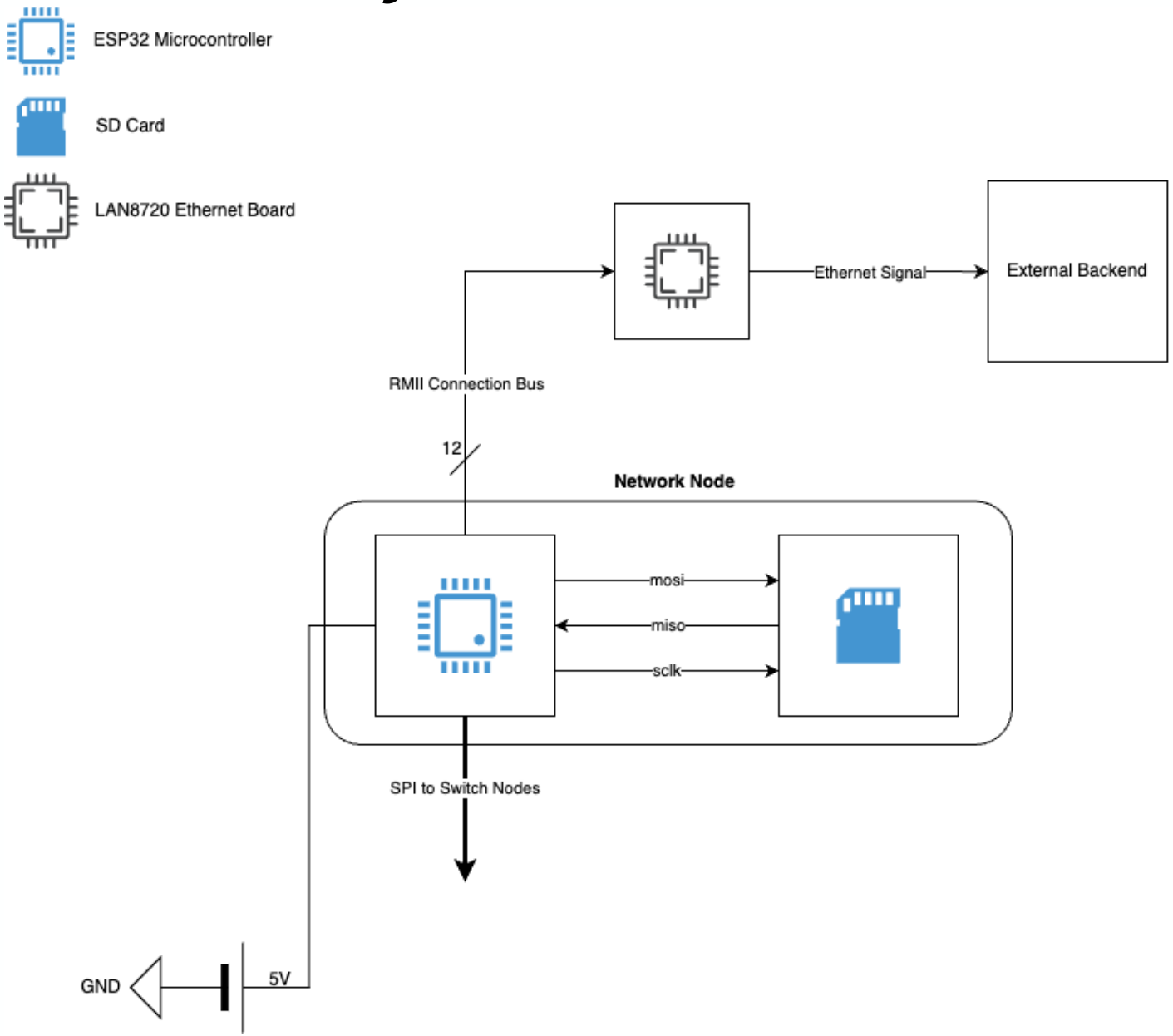
User-Interface & Back-End Layer

Specification Evidence



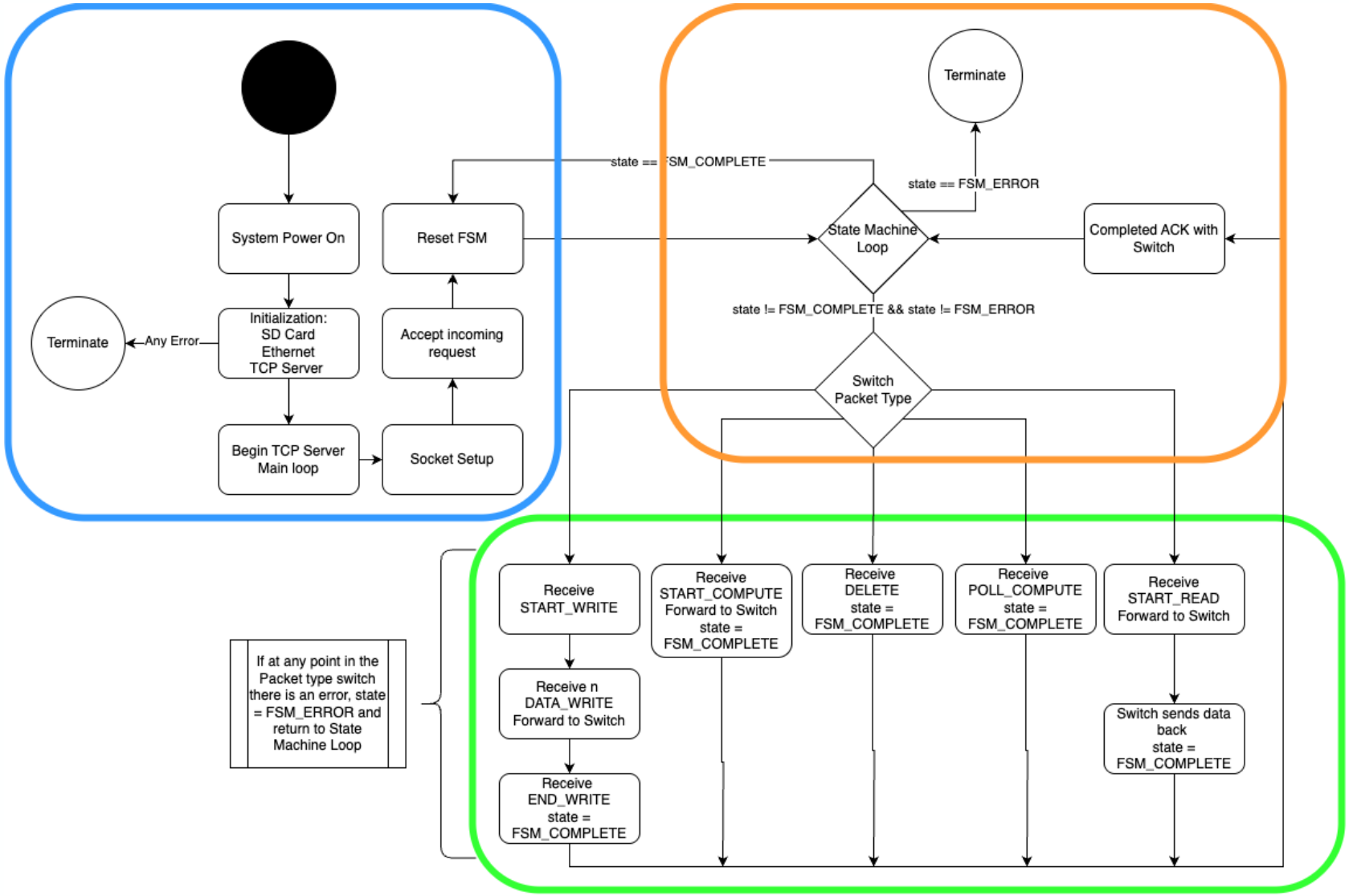
Network Coordination Layer

Block Diagram



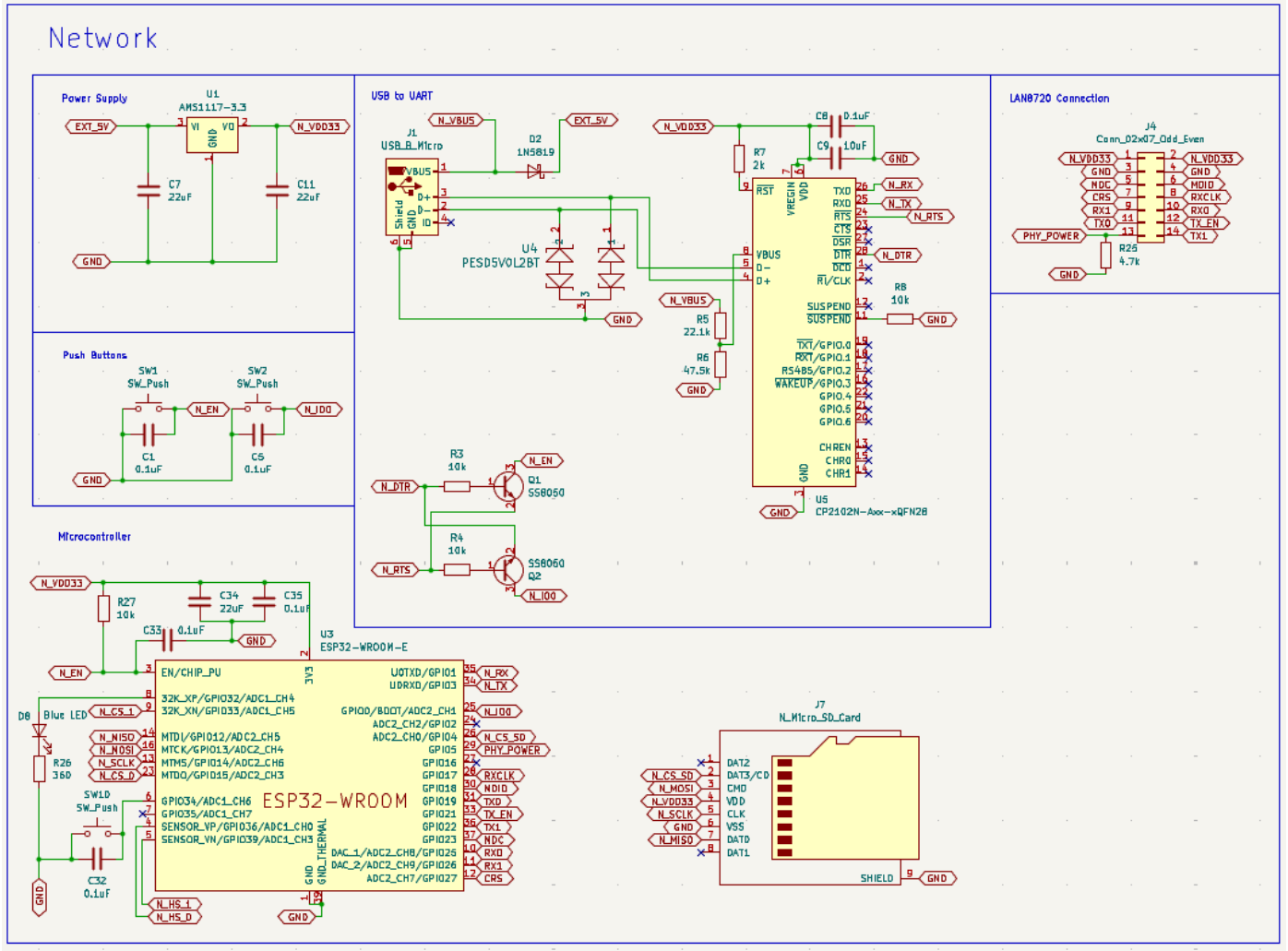
Network Coordination Layer

Activity Diagram



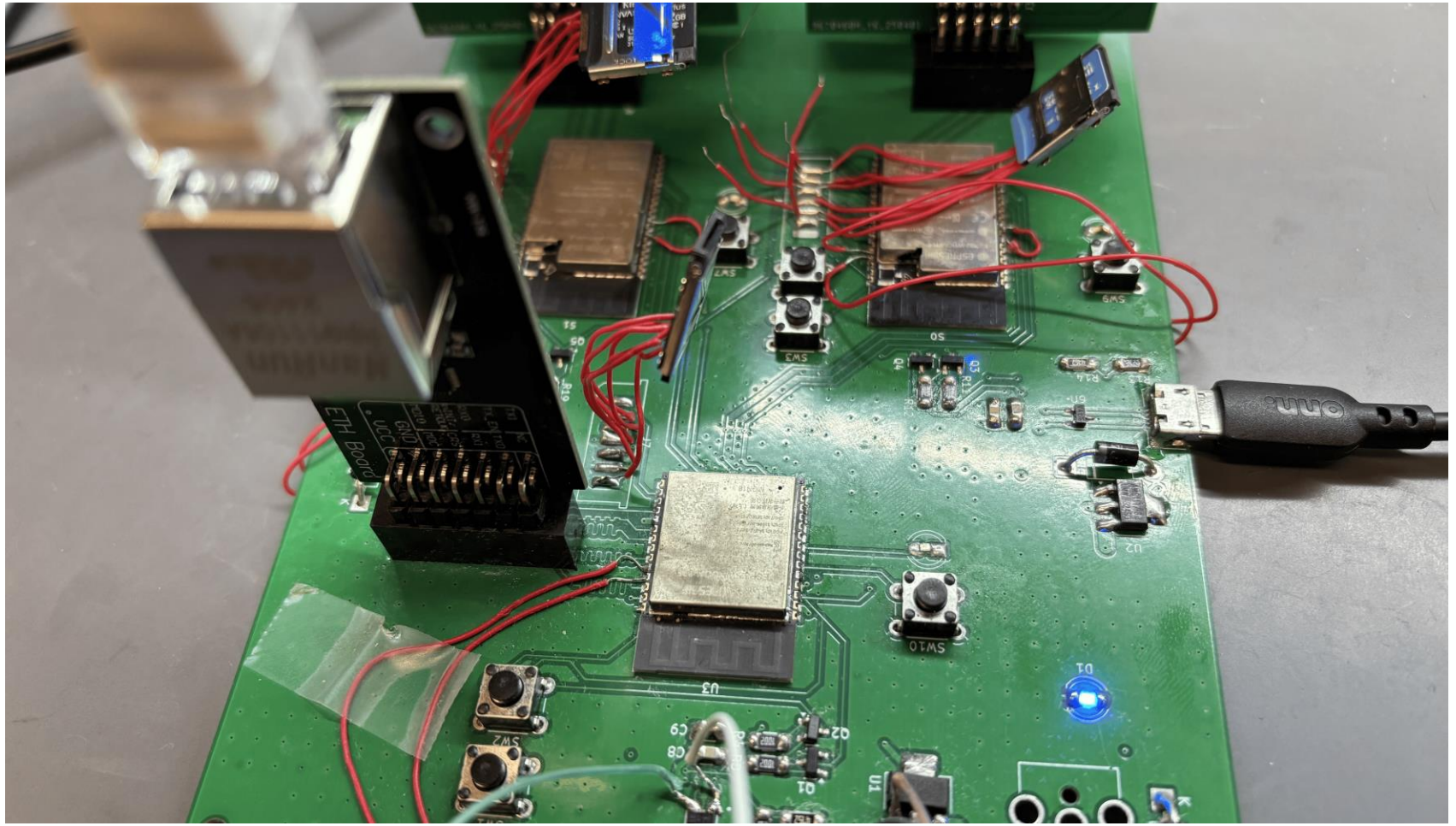
Network Coordination Layer

Schematic



Network Coordination Layer

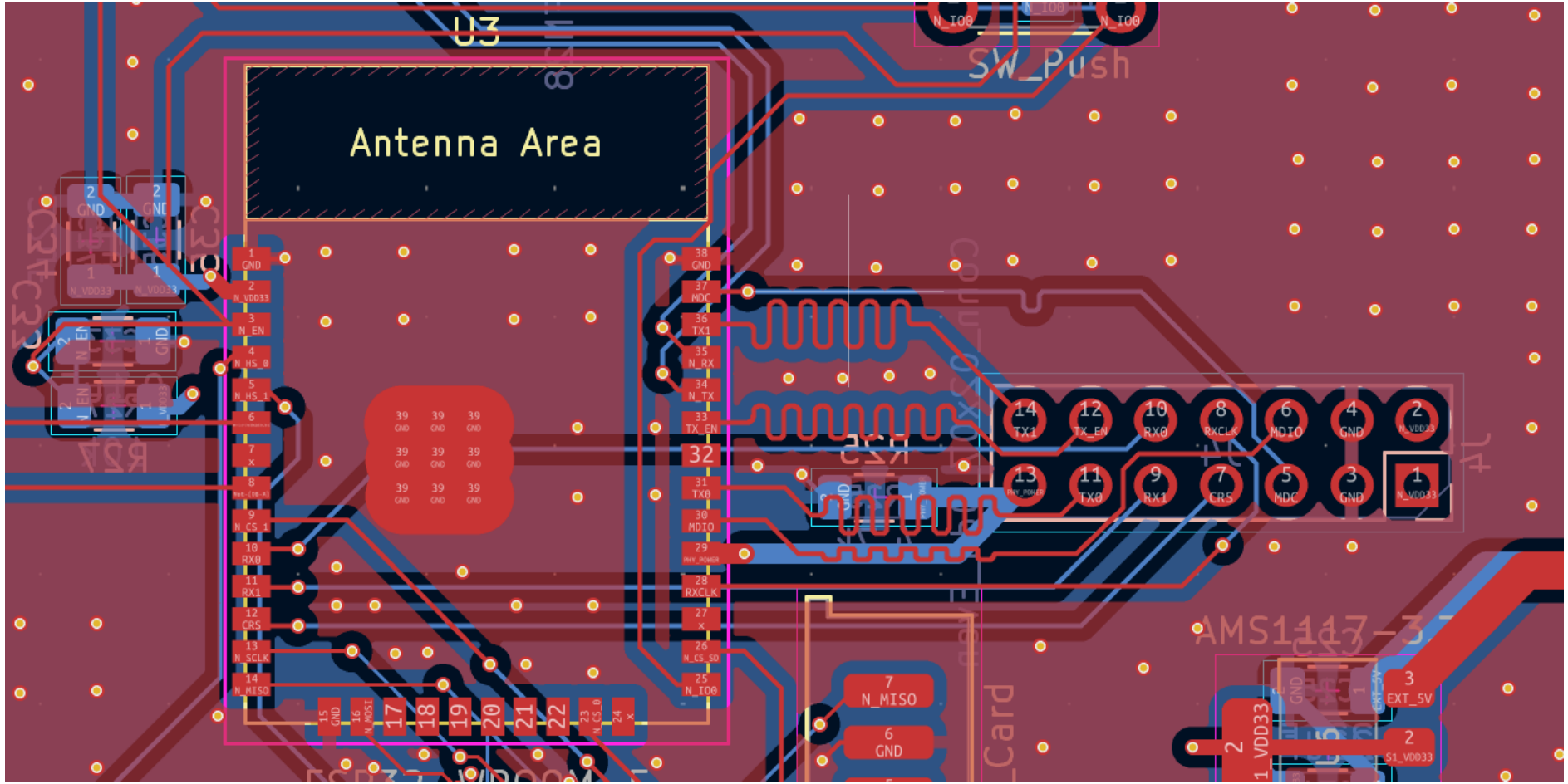
PCB



PURDUE UNIVERSITY

Network Coordination Layer

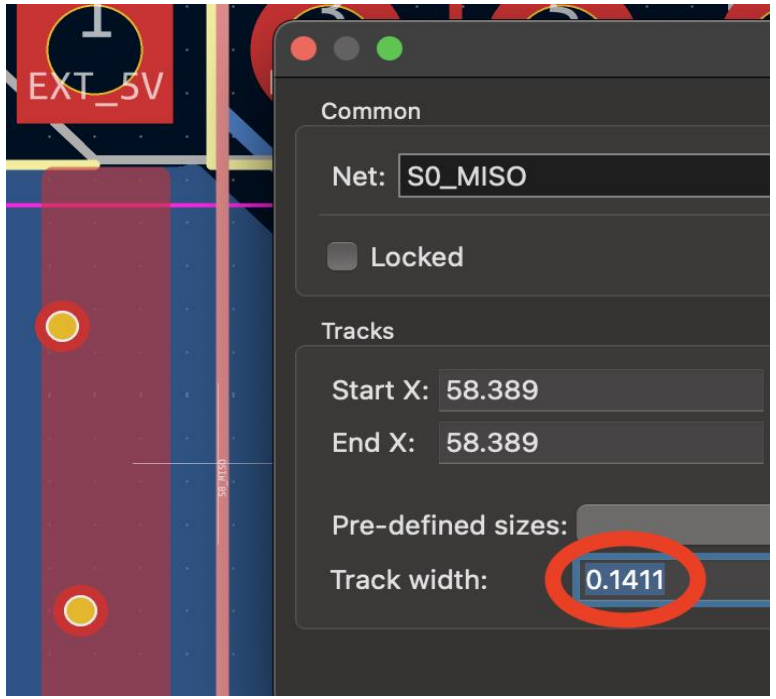
PCB

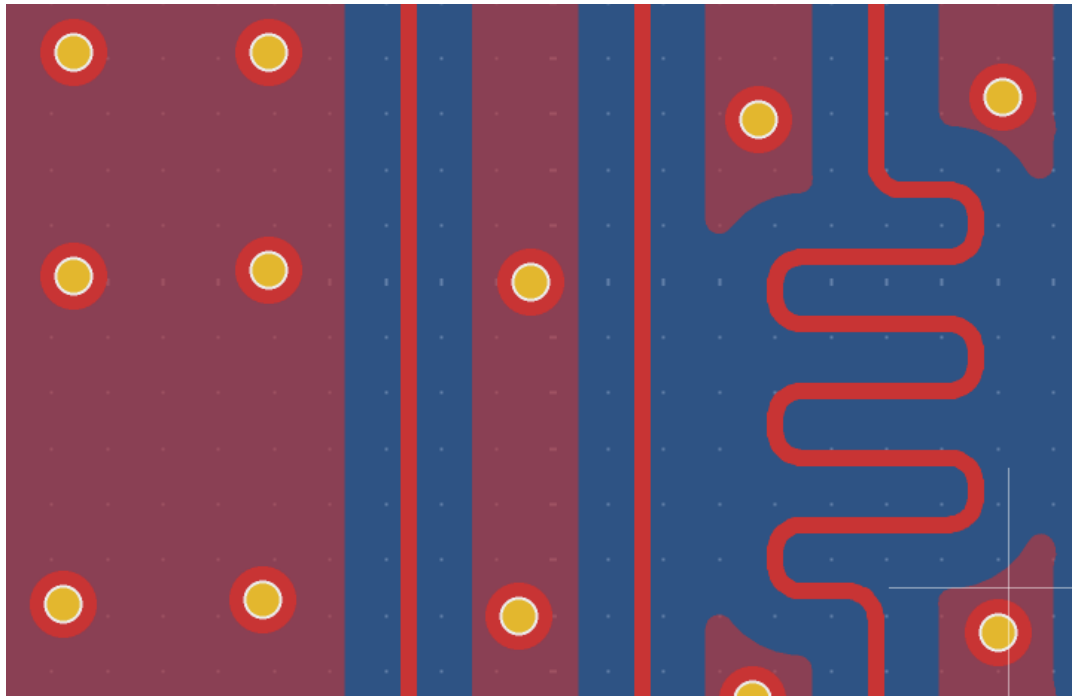


Network Coordination Layer

PCB

- Length Matching
- Impedance Matching
- Via Stitching





PURDUE UNIVERSITY

Network Coordination Layer

Specification Relevance and Justification

- Average Response Time from the network for any Transaction should be less than 100ms.
- Average Response time for a full packet should be less than 2 seconds
- The Network Node must write to the SD Card at 20 Mhz

Specification Testability

- We can test the Response time for a transaction from the backend side, timing the decision time and response time of the Network Node.
- We can time the full packet time for read and write operations by timing the system from the backend
- We can test the sd card write speed by using an oscilloscope.

Specification Motivation

- The Backend should not be held up by exceedingly long processing times on the network node
- The write and read times should be independent of compute, and quick.
- Since BoilerNet is partly a storage solution, guaranteeing a minimum write speed to the sd card is needed.

PURDUE UNIVERSITY

Specification 1 Measurement

Average Response Time from the Network Node for any Transaction should be less than 100ms.

```
I (86984) TCP_SOCKET: IN PACKET_TYPE_DATA_WRITE, data_len: 2030
I (86994) TCP_SOCKET: Sending buffer: 0xA1, 0x00 0x07 0xEE ... 0x00 0x00
I (87004) TCP_SOCKET: Received buffer: 0xA1, 0x00 0x07 0xEE ... 0x00 0x00
I (87004) TCP_SOCKET:     MATCH: SPI COMMAND: expected 0xA1, got 0xA1

Elapsed: 24929 µs
I (87014) TCP_SOCKET: IN PACKET_TYPE_DATA_WRITE, data_len: 2030
I (87334) TCP_SOCKET: Sending buffer: 0xA1, 0x00 0x07 0xEE ... 0x00 0x00
I (87344) TCP_SOCKET: Received buffer: 0xA1, 0x00 0x07 0xEE ... 0x00 0x00
I (87344) TCP_SOCKET:     MATCH: SPI COMMAND: expected 0xA1, got 0xA1

Elapsed: 333361 µs
I (87344) TCP_SOCKET: IN PACKET_TYPE_DATA_WRITE, data_len: 2030
I (87354) TCP_SOCKET: Sending buffer: 0xA1, 0x00 0x07 0xEE ... 0x00 0x00
I (87364) TCP_SOCKET: Received buffer: 0xA1, 0x00 0x07 0xEE ... 0x00 0x00
I (87364) TCP_SOCKET:     MATCH: SPI COMMAND: expected 0xA1, got 0xA1

Elapsed: 24901 µs
I (87374) TCP_SOCKET: !!!!~MID packet received~!!!!
I (87494) TCP_SOCKET: Sending buffer: 0xA2, 0x01 0x07 0xEE ... 0x00 0x00
I (87504) TCP_SOCKET: Received buffer: 0xA1, 0x00 0x07 0xEE ... 0x00 0x00
I (87504) TCP_SOCKET:     MATCH: SPI COMMAND: expected 0xA1, got 0xA1

Elapsed: 132359 µs
I (87514) TCP_SOCKET: IN PACKET_TYPE_DATA_WRITE, data_len: 2030
I (87514) TCP_SOCKET: Sending buffer: 0xA1, 0x00 0x07 0xEE ... 0x00 0x00
I (87524) TCP_SOCKET: Received buffer: 0xA2, 0x01 0x07 0xEE ... 0x00 0x00
I (87524) TCP_SOCKET:     MATCH: SPI COMMAND: expected 0xA2, got 0xA2

Elapsed: 20227 µs
I (87534) TCP_SOCKET: IN PACKET_TYPE_DATA_WRITE, data_len: 2030
I (87544) TCP_SOCKET: Sending buffer: 0xA1, 0x00 0x07 0xEE ... 0x00 0x00
I (87554) TCP_SOCKET: Received buffer: 0xA1, 0x00 0x07 0xEE ... 0x00 0x00
I (87554) TCP_SOCKET:     MATCH: SPI COMMAND: expected 0xA1, got 0xA1
```

```
I (8104) TCP_SOCKET: Received buffer: 0xB1, 0x07 0xFD 0x7D ... 0x0A 0x20
I (8104) TCP_SOCKET: DATA_READ packet: data length = 2045
I (8114) TCP_SOCKET: Forwarded DATA_READ packet, 2045 bytes
Elapsed: 26921 µs
I (8124) TCP_SOCKET:     Receiving Packet
I (8194) TCP_SOCKET: Sending buffer: 0xB1, 0x01 0x00 0x00 ... 0x00 0x00
I (8204) TCP_SOCKET: Received buffer: 0xB1, 0x07 0xFD 0x20 ... 0x28 0x6E
I (8204) TCP_SOCKET: DATA_READ packet: data length = 2045
I (8204) TCP_SOCKET: Forwarded DATA_READ packet, 2045 bytes
Elapsed: 89073 µs
I (8214) TCP_SOCKET:     Receiving Packet
I (8214) TCP_SOCKET: Sending buffer: 0xB1, 0x01 0x00 0x00 ... 0x00 0x00
I (8224) TCP_SOCKET: Received buffer: 0xB1, 0x07 0xFD 0x61 ... 0x66 0x6F
I (8224) TCP_SOCKET: DATA_READ packet: data length = 2045
I (8234) TCP_SOCKET: Forwarded DATA_READ packet, 2045 bytes
Elapsed: 26902 µs
I (8234) TCP_SOCKET:     Receiving Packet
I (8314) TCP_SOCKET: Sending buffer: 0xB1, 0x01 0x00 0x00 ... 0x00 0x00
I (8324) TCP_SOCKET: Received buffer: 0xB1, 0x07 0xFD 0x72 ... 0x6E 0x67
I (8324) TCP_SOCKET: DATA_READ packet: data length = 2045
I (8324) TCP_SOCKET: Forwarded DATA_READ packet, 2045 bytes
Elapsed: 89090 µs
I (8334) TCP_SOCKET:     Receiving Packet
I (8334) TCP_SOCKET: Sending buffer: 0xB1, 0x01 0x00 0x00 ... 0x00 0x00
I (8344) TCP_SOCKET: Received buffer: 0xB1, 0x07 0xFD 0x5F ... 0x0A 0x20
I (8344) TCP_SOCKET: DATA_READ packet: data length = 2045
I (8354) TCP_SOCKET: Forwarded DATA_READ packet, 2045 bytes
Elapsed: 26918 µs
I (8354) TCP_SOCKET:     Receiving Packet
I (8434) TCP_SOCKET: Sending buffer: 0xB1, 0x01 0x00 0x00 ... 0x00 0x00
I (8434) TCP_SOCKET: Received buffer: 0xB1, 0x07 0xFD 0x20 ... 0x61 0x22
I (8434) TCP_SOCKET: DATA_READ packet: data length = 2045
I (8444) TCP_SOCKET: Forwarded DATA_READ packet, 2045 bytes
```

PURDUE UNIVERSITY

Specification 2 Measurement

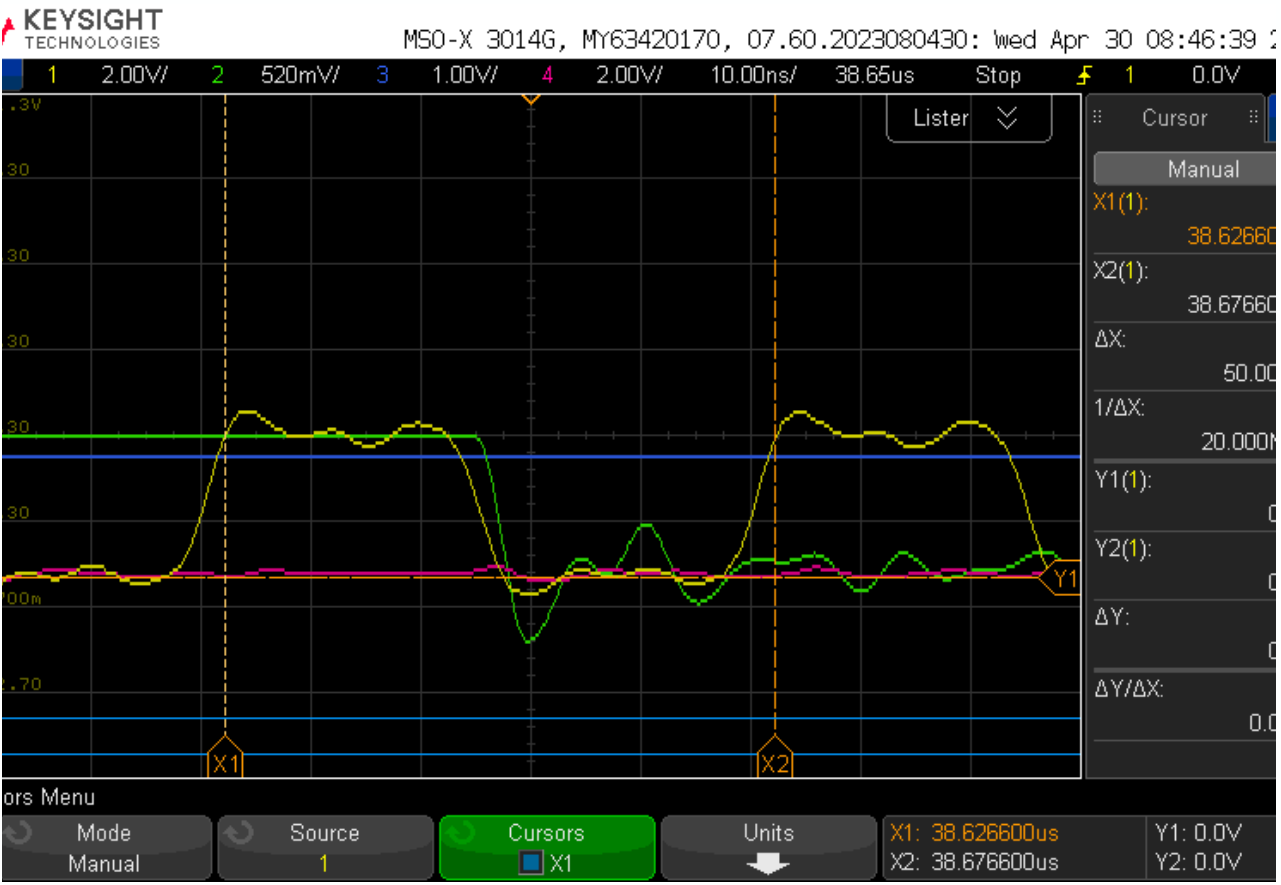
Average Response Time for a full packet should be less than 2 s.

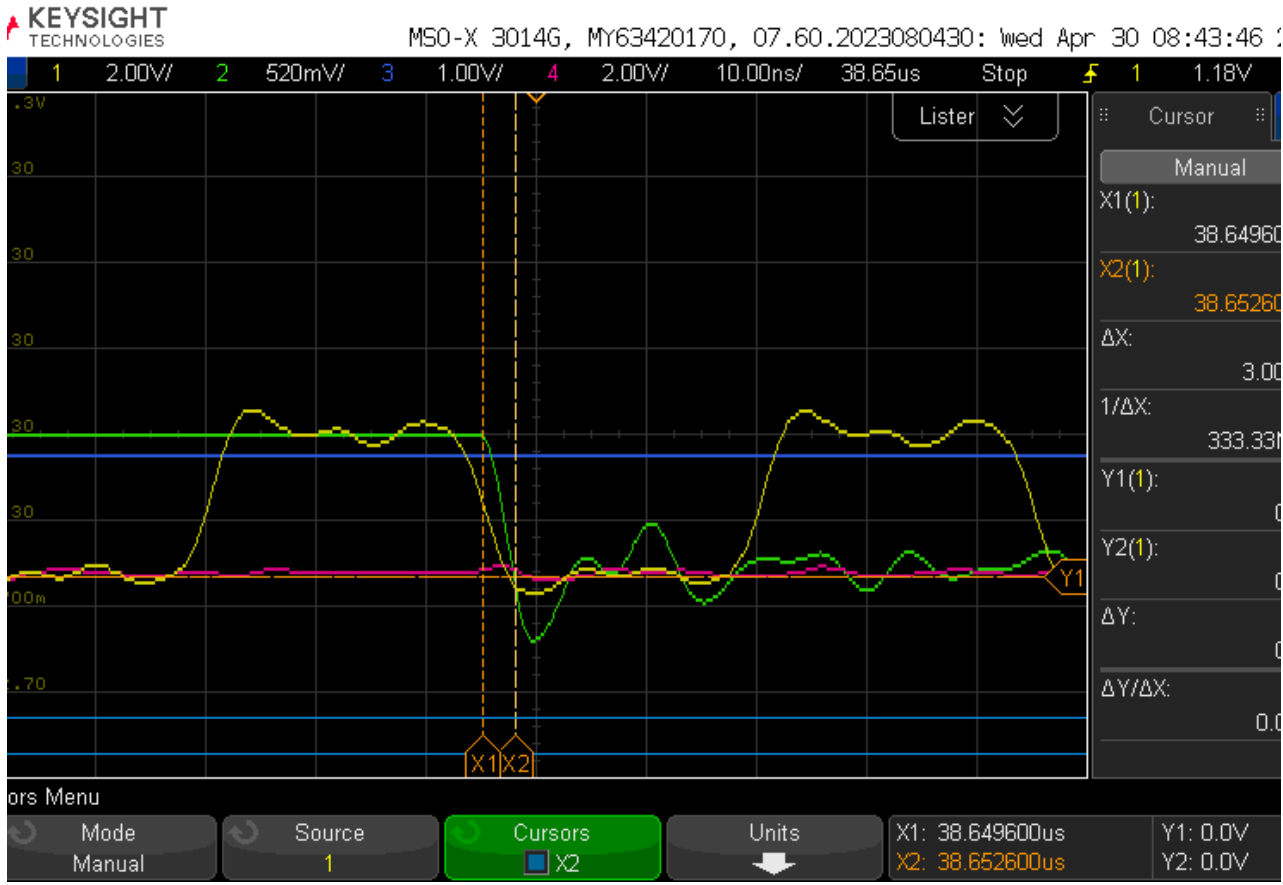
```
(base) root@araviki:/home/araviki/workbench/boilernet/test# python3 test_routines.py 8080 192.168.0.50 chiahua.jpg read user02
OPERATION: read | USER_ID: user02
Connecting over Socket
Opened Socket!
READ complete, checksum=00AD9339

=== Timing Summary ===
Total: 2994.49 ms
Avg pkt time: 7.36 ms
  ACK_B0: 18.77 ms
  DATA READ: 7.10 ms
```

```
(base) root@araviki:/home/araviki/workbench/boilernet/test# python3 test_routines.py 8080 192.168.0.50 chiahua.jpg write user02
OPERATION: write | USER_ID: user02
Connecting over Socket
Opened Socket!
STORE on FILENAME: ./input/chiahua.jpg | ext: .jpg
Total Bytes: 89604 | Packets: 45
Sent END packet (write) with checksum 0x00ad9339
WRITE complete

=== Timing Summary ===
Total: 4486.31 ms
Avg pkt time: 1478.80 ms
  ACK_A0: 12.25 ms
  ACK_A2: 2749.31 ms
  ACK_A3: 1674.84 ms
```

Specification 3 Measurement

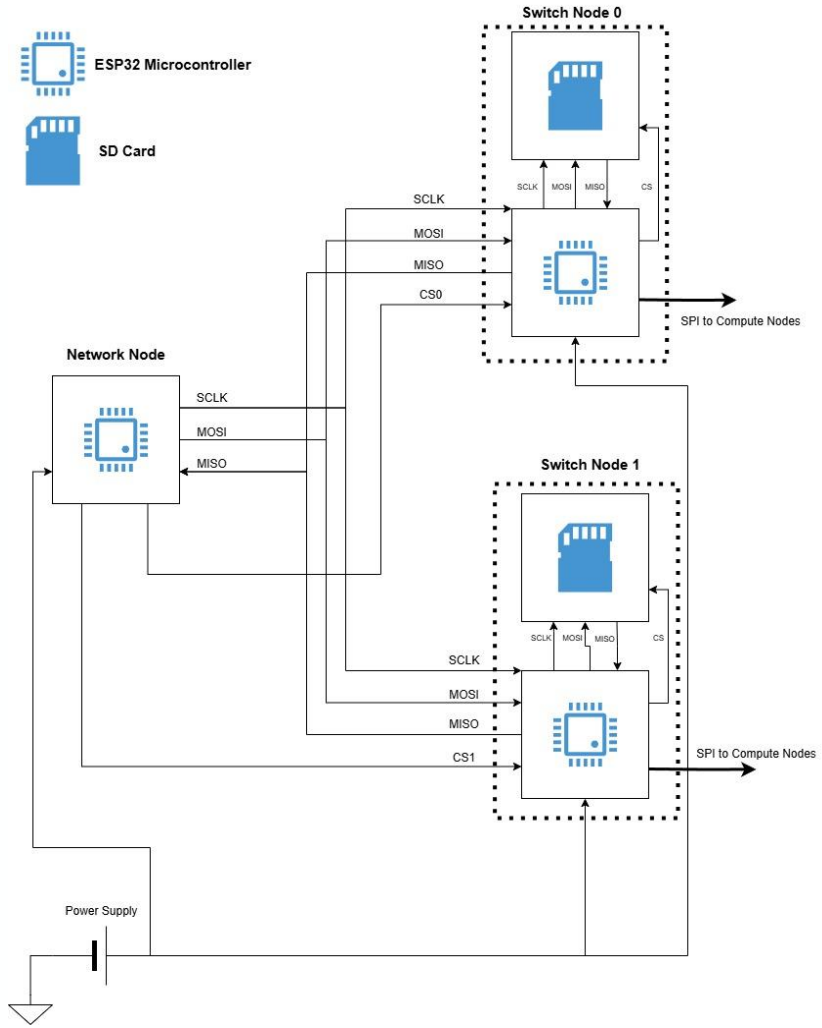
The Network Node must write to the SD Card at 20 Mhz





Internal Switch Routing Layer

Block Diagram



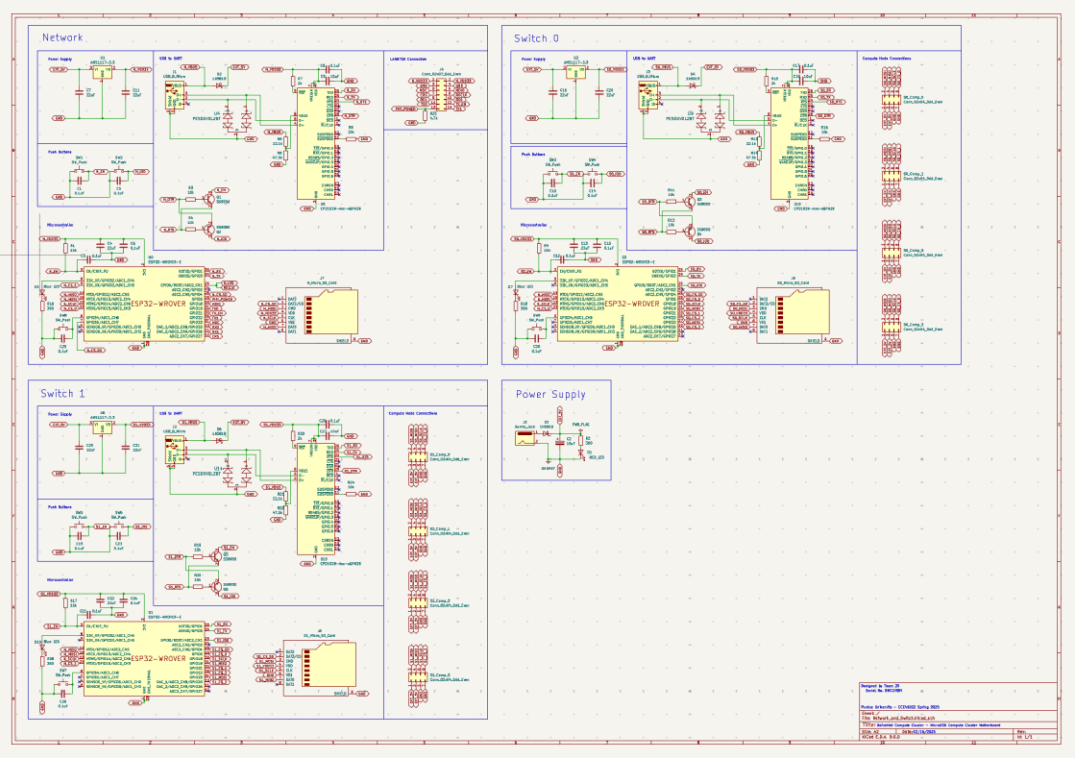
PURDUE UNIVERSITY

Internal Switch Routing Layer

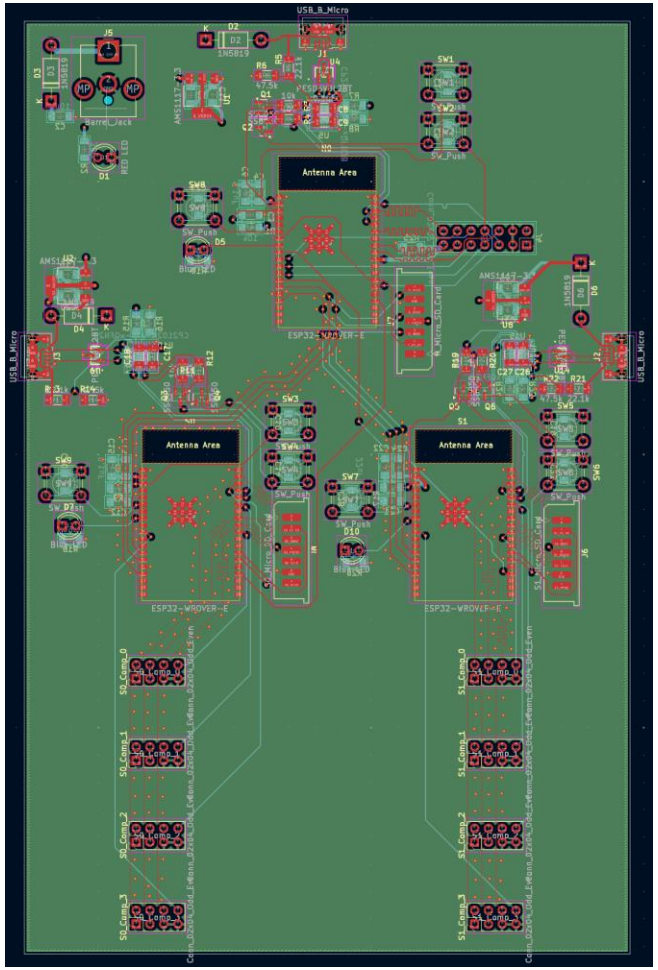
Schematic

Network

Switch 1



Switch 0

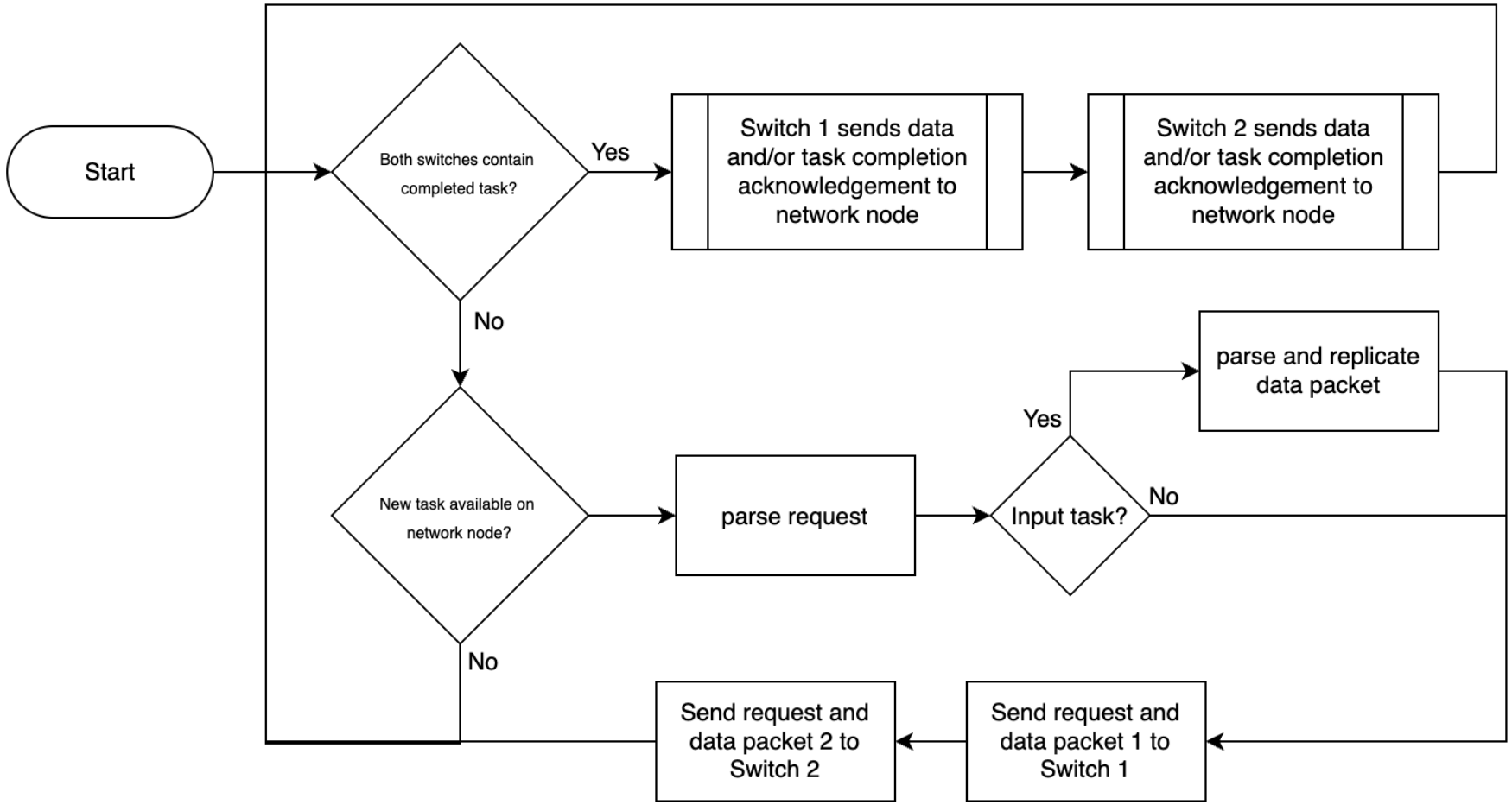


Each Node has a similar schematic

PURDUE UNIVERSITY

Internal Switch Routing Layer

Activity Diagram



PURDUE UNIVERSITY

Internal Switch Routing Layer

Specification Relevance and Justification

- **Relevance**: Groups the compute nodes using switches to allow for fewer slave devices per communication master and allows data placement over multiple storage devices.
- **Justification**: Allowing the cluster to store data across different storage mediums on different nodes allows for much greater file storage. Utilizing two different switch nodes also lets the cluster scale more effectively and minimize congestion on any one communication bus.

Specification Testability

- Speed of subsystem and subsystem parts can be tested by measuring the time it takes between a command and data being received by the network node and the second switch node ready to send the data to its compute nodes.
- Check to see if error packets are being received successfully
- Check for data persistence
- Check for critical timing delay with a similar method to subsystem speed measurement

Specification Motivation

- The Internal Switch Routing Layer was devised in order to add a hierarchical level to the compute cluster for boosting cluster scalability and for introducing data storage striping across multiple storage mediums.

Specifications

Blue – MET, Yellow – PARTIALLY MET, Red - NOT MET

- Individual packets can be received and sent at greater than 50KBps over SPI.
- The Critical Path Delay in a Switch -- from receiving a packet, to preparing for the next one -- must be under 200ms.
- The Switch Nodes must be able to communicate errors to both the Network Node and Compute Nodes.
- The Switch Nodes must maintain data when there is a power loss.

Specification Measurement

Individual packets can be received and sent at greater than 50KBps over

```
[recv] DATA_READ_LEN: 2045 bytes -> 0.006s
[recv] DATA_READ_LEN: 2045 bytes -> 0.005s
[recv] DATA_READ_LEN: 1531 bytes -> 0.005s
[recv] END_READ_CHECKSUM: 0x00 0x83 0xCE 0xC0
READ complete
  Total bytes: 66971
  Duration:    2.518s
  Throughput (including I/O + Control Logic):  26600.51 bytes/s
  Checksum:    0083CEC0

=== Timing Summary ===
Total: 2548.58 ms
Avg pkt time: 9.15 ms
  ACK_B0: 22.13 ms
  DATA_READ: 8.76 ms
```

26.6 KB/s

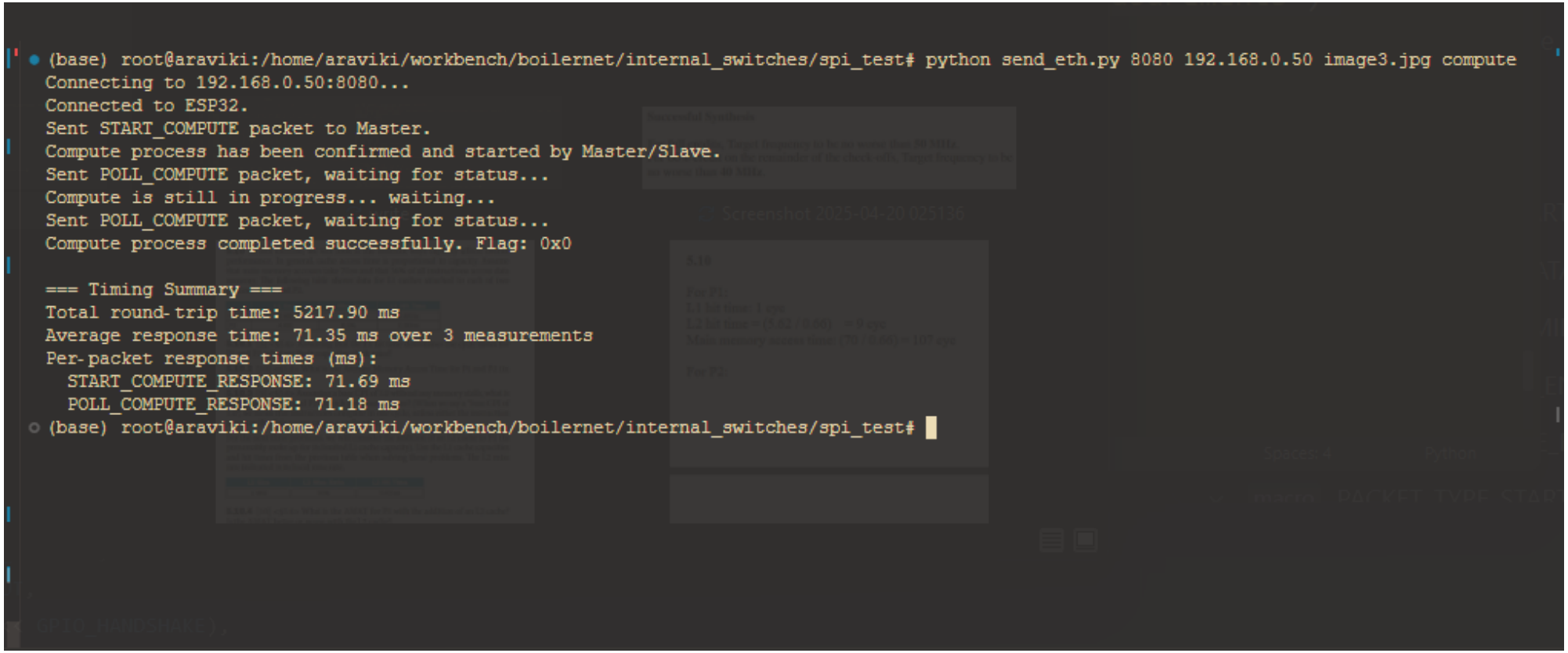
```
[recv] DATA_READ_LEN: 2030 bytes -> 0.000s
[recv] DATA_READ_LEN: 2011 bytes -> 0.000s
Sent END packet (write) with checksum 0x0083cec0
WRITE complete
  Duration:   3.623s
  Throughput: 18487.04 bytes/s

=== Timing Summary ===
Total: 3623.03 ms
Avg pkt time: 100.59 ms
  ACK_A0: 100.51 ms
  DATA_WRITE: 0.01 ms
  ACK_A2: 2213.27 ms
  ACK_A3: 1307.28 ms
```

18.5 KB/s

PURDUE UNIVERSITY

Specification Measurement

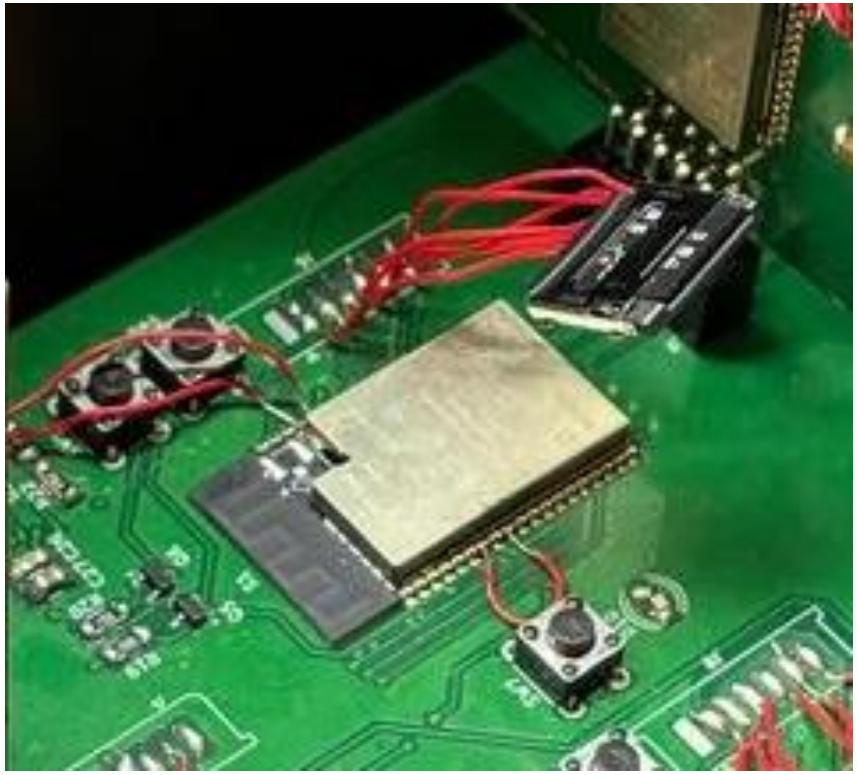
The Critical Path Delay in a Switch – from receiving a packet, to preparing for the next one – must be under 200ms.



Specification Measurement

```
(base) root@araviki:/home/araviki/workbench/boilernet/internal_switches/s
age3.jpg write
Connecting to 192.168.0.50:8080...
Connected to ESP32.
Image loaded: 25x25, total 9155 bytes
num_full_packets  4
remainder  1035
total_packets  5
mid_packet_index  2
Sent START packet for write.
ACK received: 160
Received ACK for START (write).
Sent MID packet (write).
ACK received: 162
Received ACK for MID (write).
Sent END packet (write) with checksum 0x0011892c
ACK received: 164
Got PACKET_TYPE_FUC
FSM (write): Transfer encountered an error.

=== Timing Summary ===
Total round-trip time: 1680.06 ms
Average response time: 559.52 ms over 3 measurements
Per-packet response times (ms):
  ACK_A0: 25.32 ms
  ACK_A2: 177.97 ms
  ACK_A3: 1475.28 ms
```

The Switch Nodes must be able to communicate errors to both the Network Node and Compute Nodes.

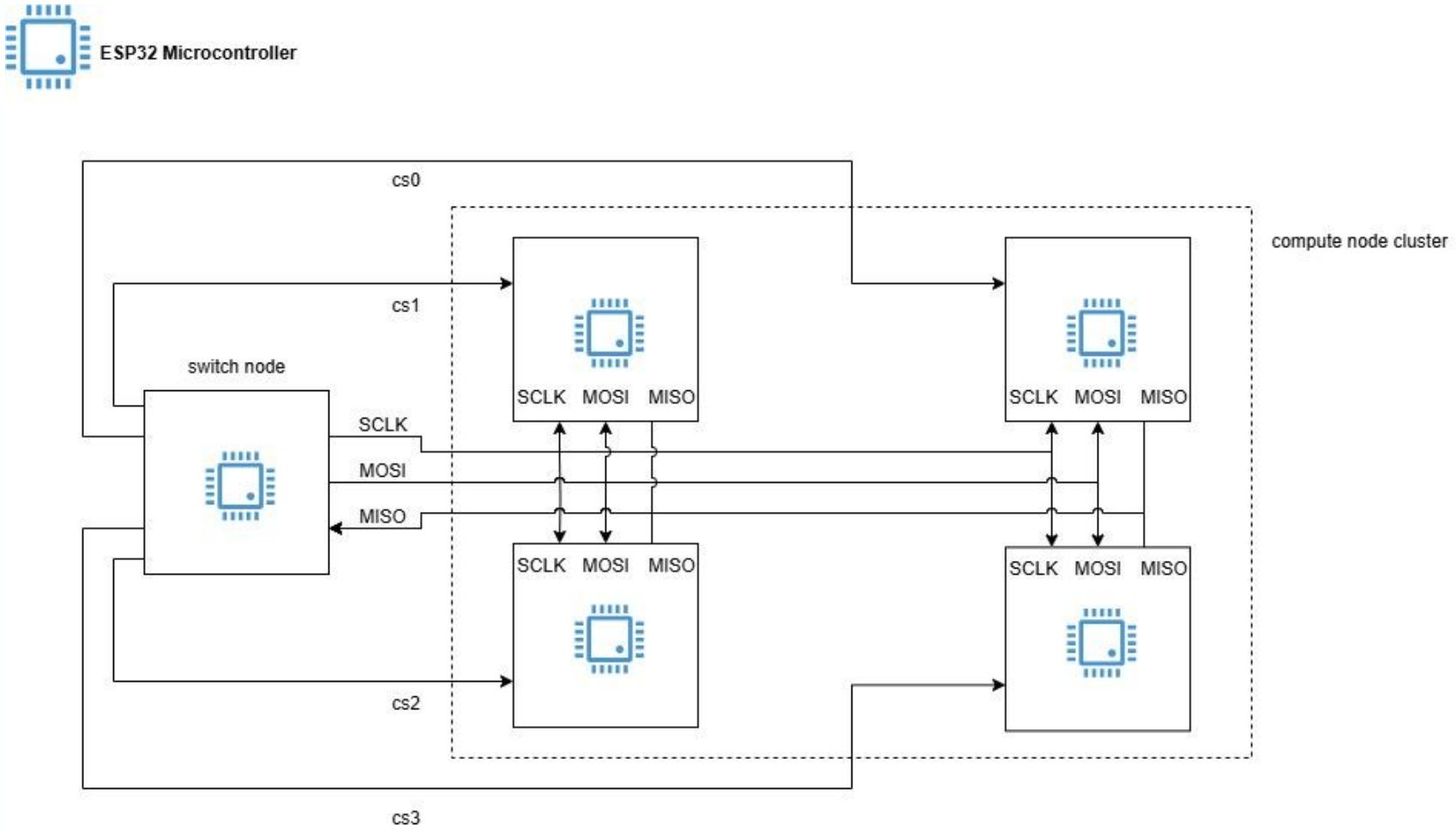


The Switch Nodes must maintain data when there is a power loss.

PURDUE UNIVERSITY

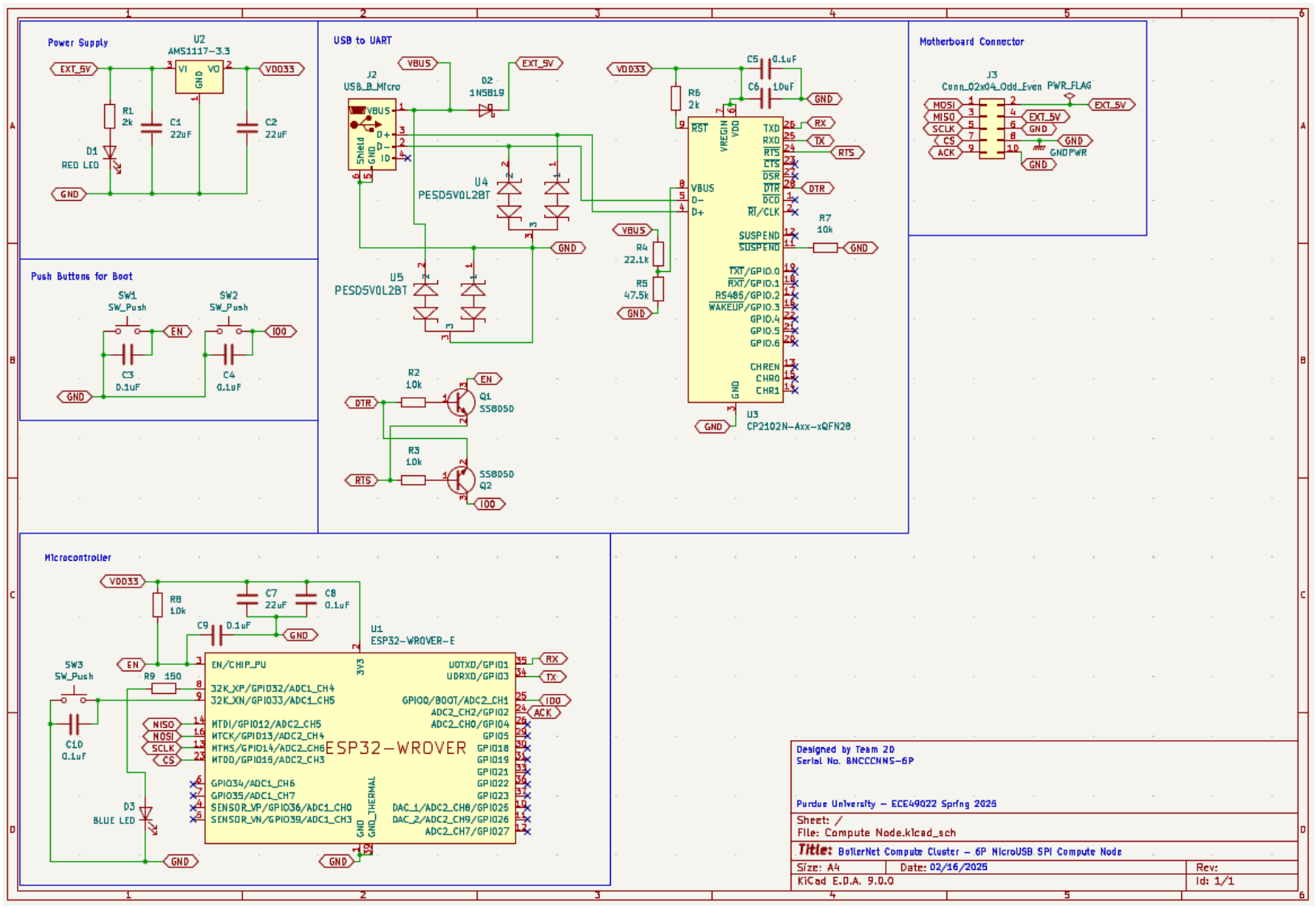
Distributed Compute Layer

Block Diagram



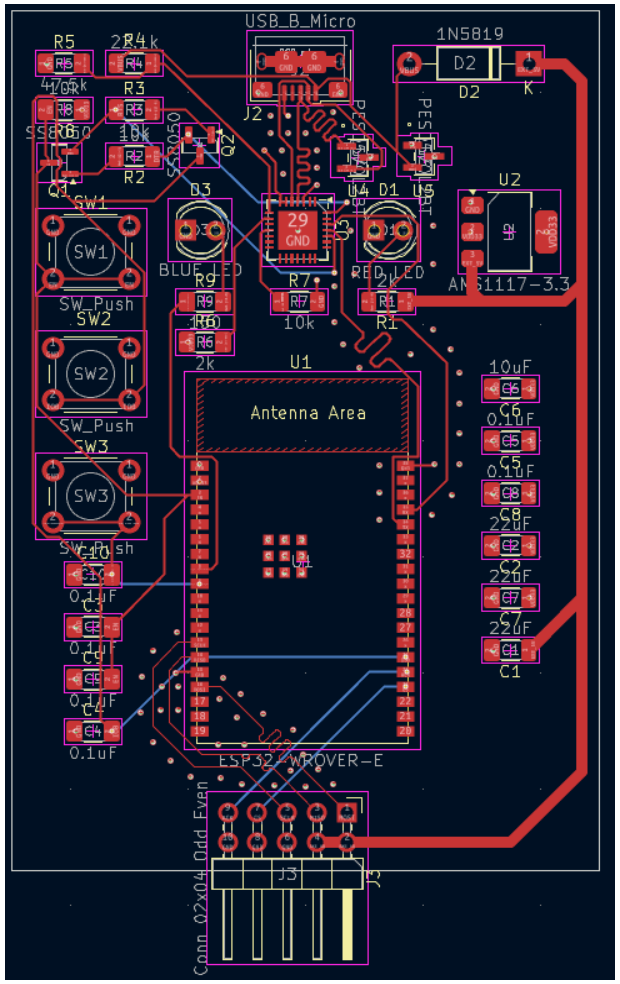
Distributed Compute Layer

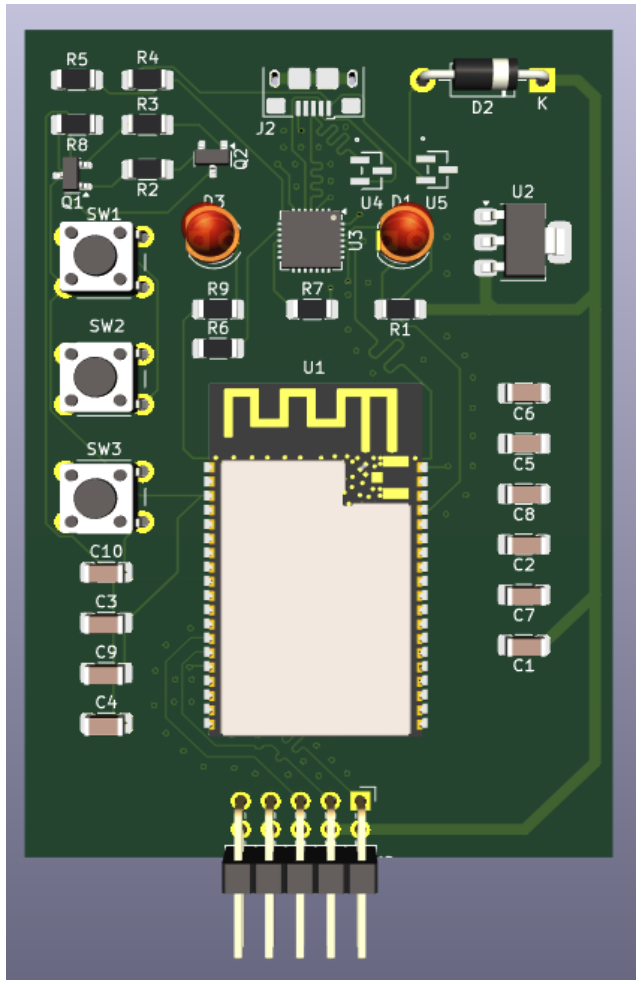
Schematic

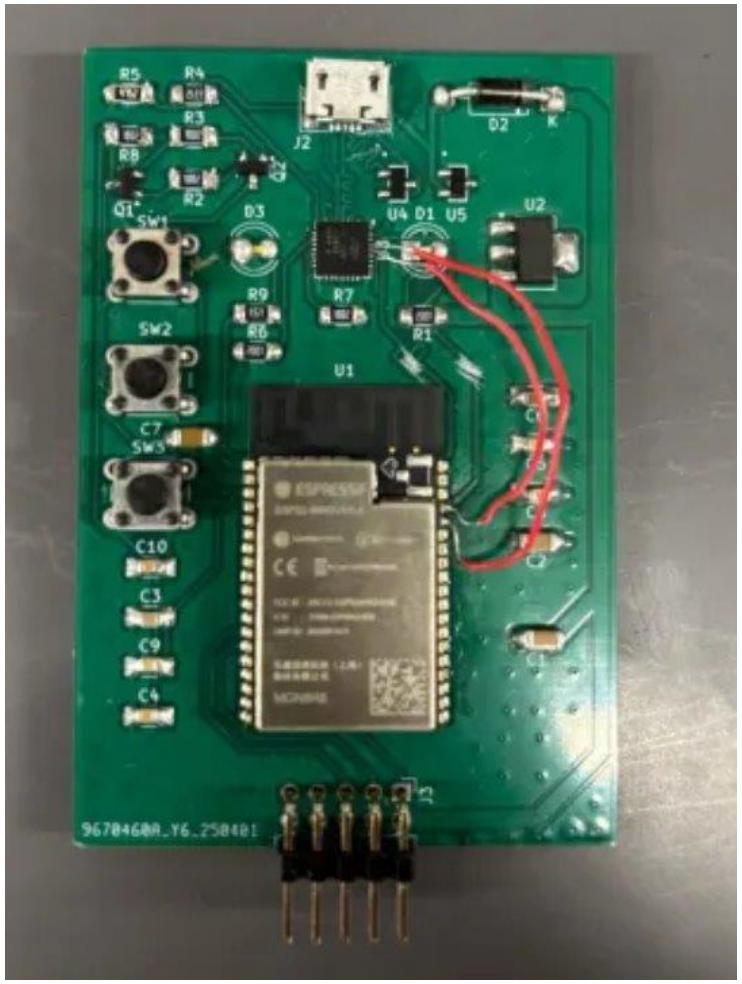


Distributed Compute Layer

PCB







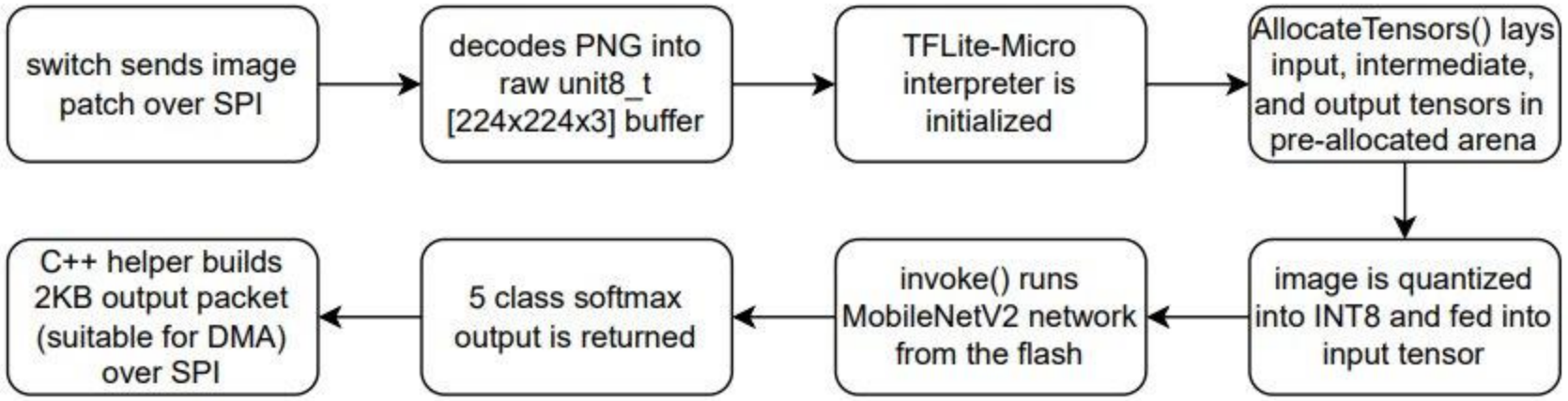
PURDUE UNIVERSITY

Distributed Compute Layer

Specifications

- Deep Learning Model running on a Compute Node must not take more than 10 seconds for inference
- Compute Nodes must decode PNG buffers into Byte Arrays in under 2s
- Compute Nodes must be able to fit the entire PNG Buffer, Quantized Activations and intermediate Buffers in RAM

Activity Diagram



Distributed Compute Layer

Specification Relevance and Justification

- **Relevance**: Offloads workload away from switch nodes and parallelizes compute among 4 different compute nodes to speed up the overall image inference tasks
- **Justification**: Switch nodes already have multiple tasks such as interfacing with SD card and network nodes and adding computation to that would slow down the task completion speed.

Specification Testability

- Speed of task completion can be tested by measuring the time it takes for the Deep Learning Model running on the Compute node for inference
- Size of task can be tested by checking size of PNG buffer sent without RAM overflow

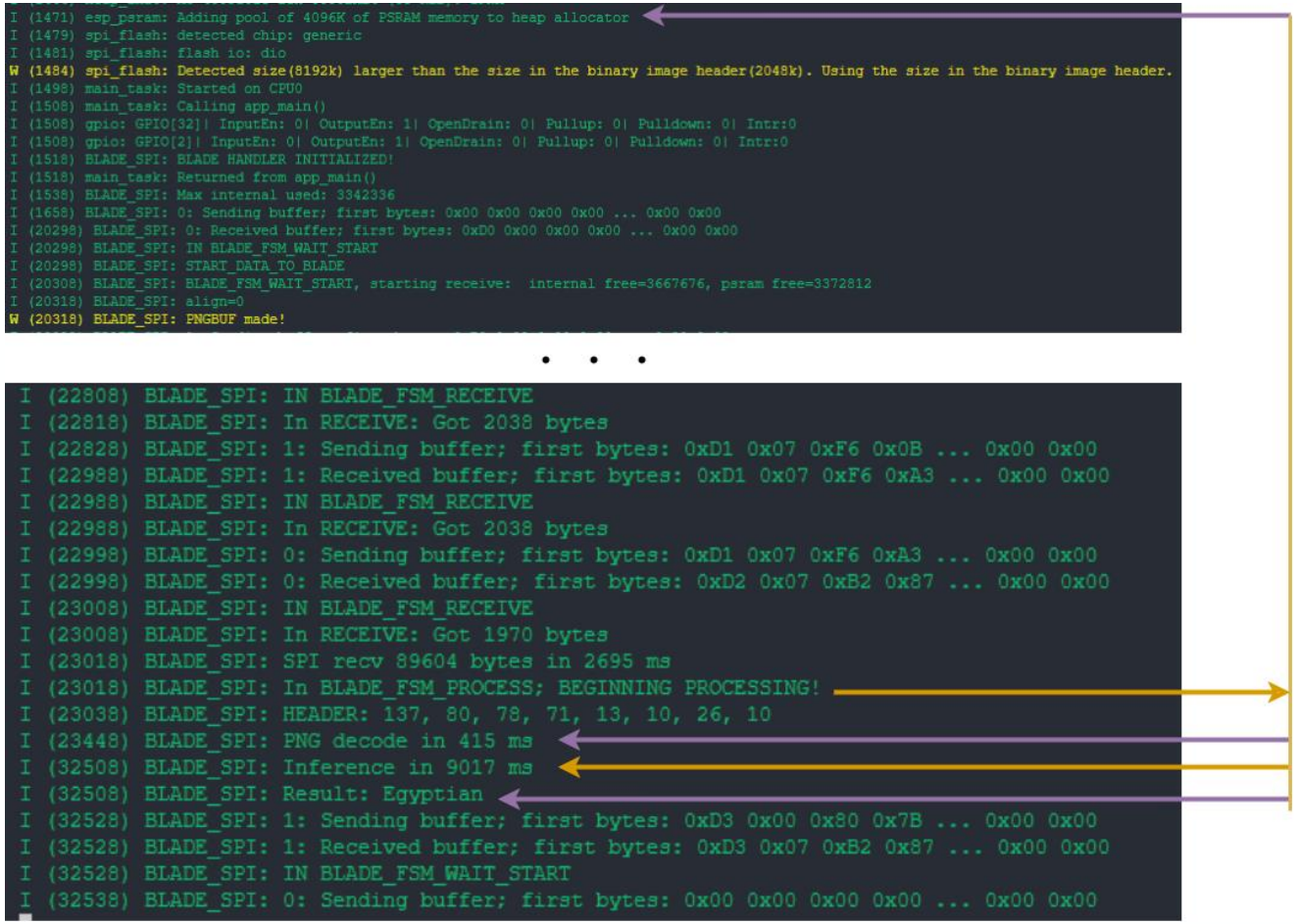
Specification Motivation

- The Distributed Compute Layer was devised in order to increase the speed of completing tasks to a reasonable level for the purposes of a compute cluster. The distributed compute layer aims to alleviate the issue of microprocessors having slower speeds by parallelizing the task

Distributed Compute Layer

Specification Evidence

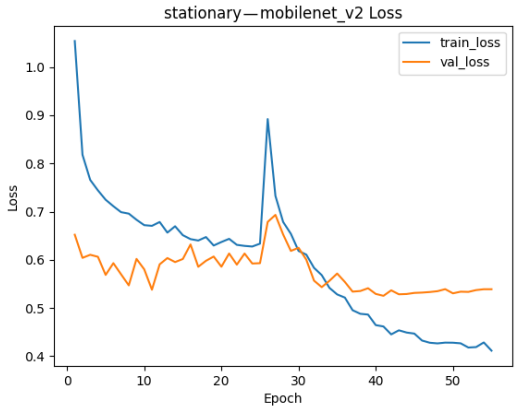
- Deep Learning Model running on a Compute Node must not take more than 10 seconds for inference
  - Inference takes 9.017 seconds as shown by the 2nd yellow arrow
- Compute Nodes must decode PNG buffers into Byte Arrays in under 2s
  - LodePNG's open-source decoder and the WROVER's SPI-RAM allowed us to override general malloc() calls to help in larger working buffers resulting in a sub-500ms decoding time shown by the 2nd purple arrow
- Compute Nodes must be able to fit the entire PNG Buffer, Quantized Activations and intermediate Buffers in RAM
  - We utilize the SPIRAM-enabled WROVER IE boards to ensure internal RAM doesn't overflow. With the 224x224 tiles taking over 300KB and the TF-Lite's Tensor Arena requiring over 500KB of working memory, internal RAM would not be enough. We ensure robustness by cancelling transactions overflowing a preset utilization limit of 4MB. This is shown by the first purple arrow showing 4MB of PSRAM being initialized.

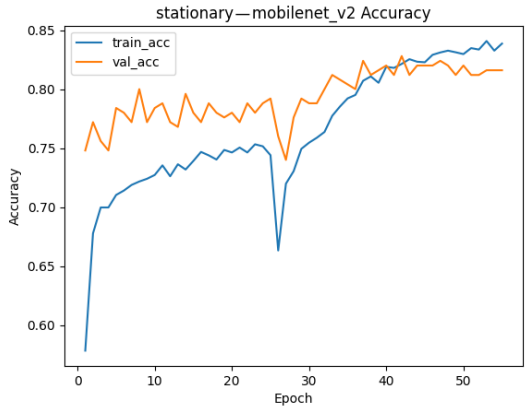


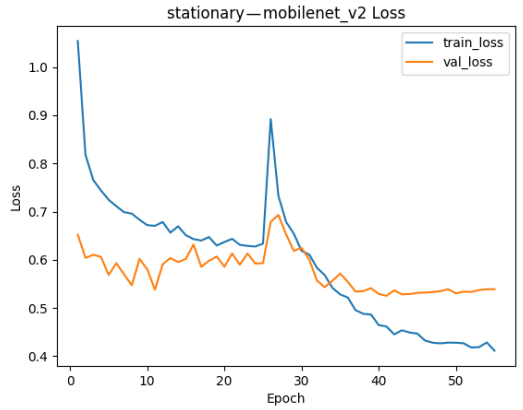
PURDUE UNIVERSITY

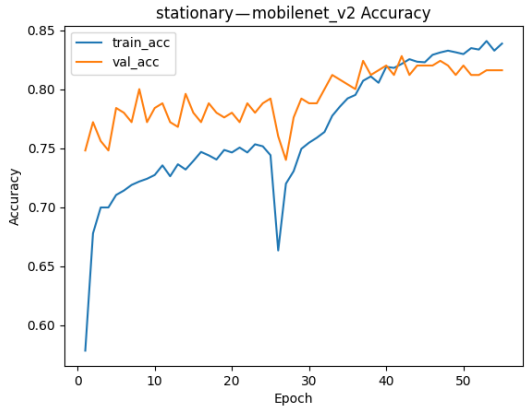
Distributed Compute Layer

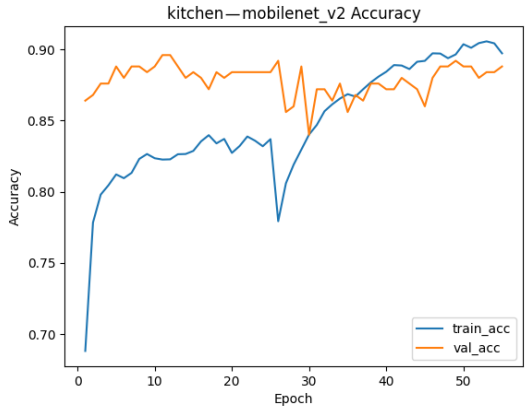
Specification Evidence

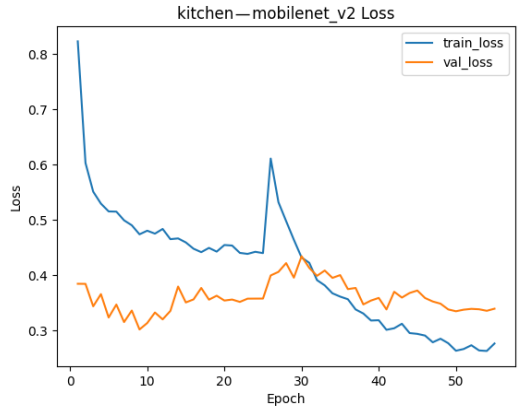


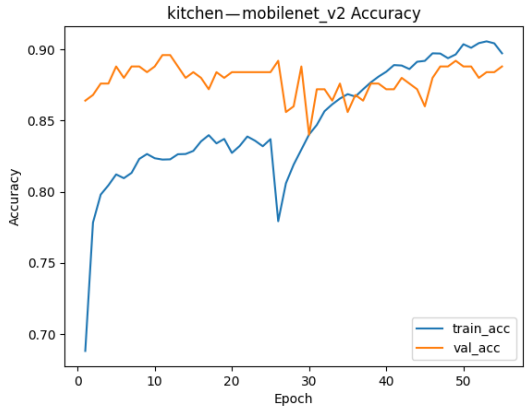










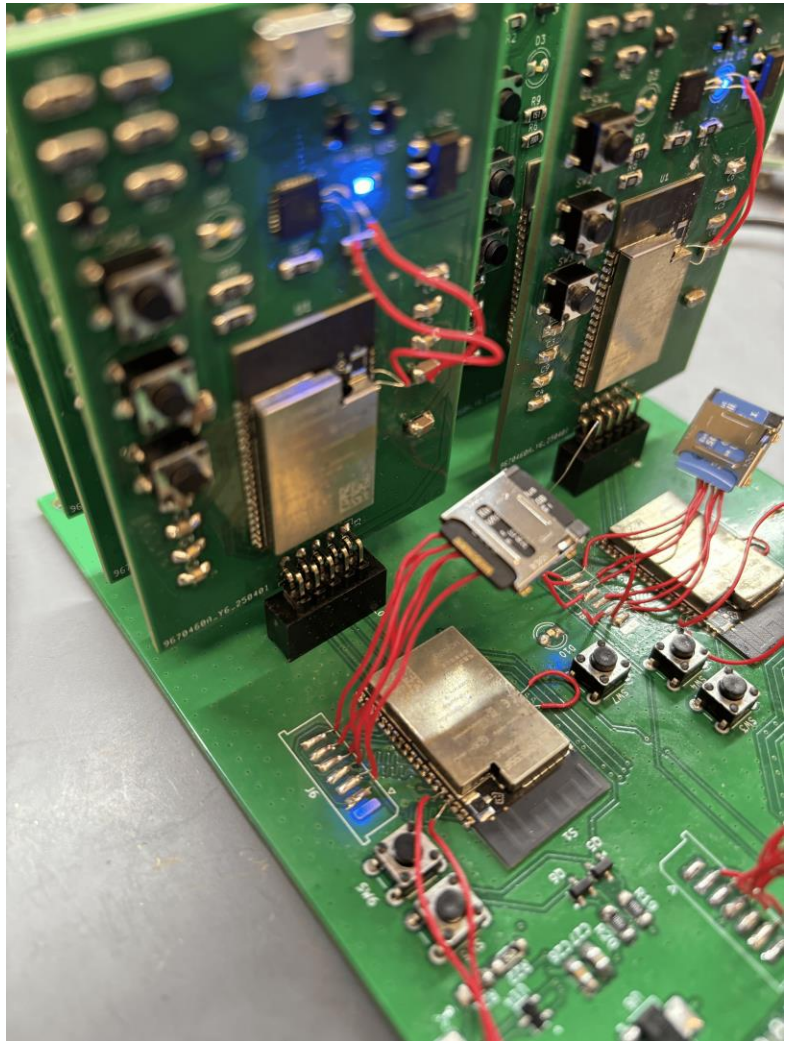


PURDUE UNIVERSITY

In-Retrospect!

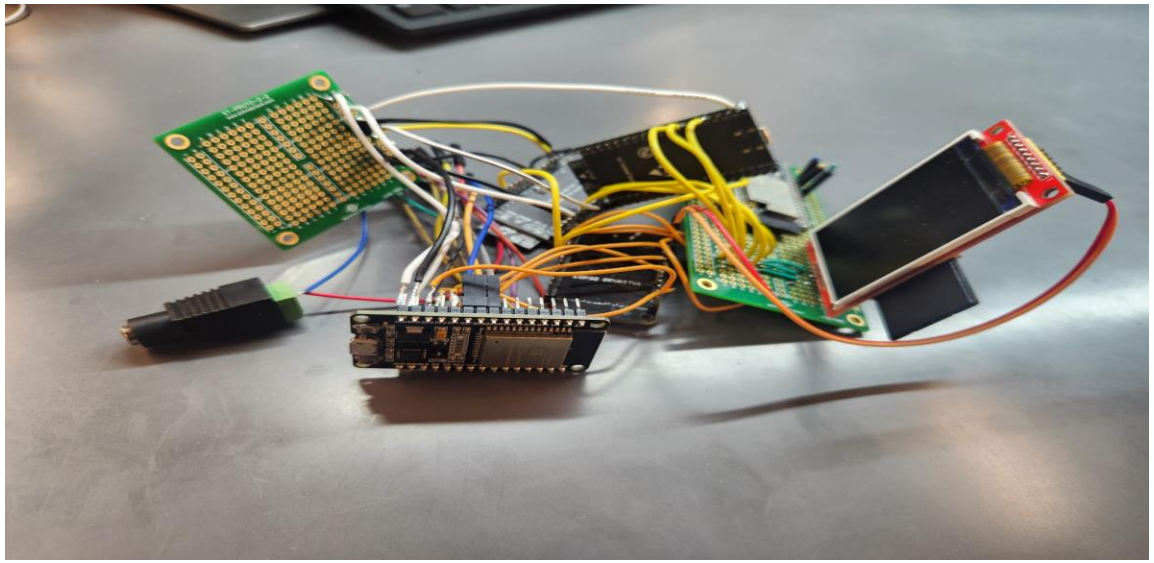
- Footprint PCB Management
  - Barrel Jack direction
  - Sd Card Reader footprint
  - Capacitor and Resistor Packages
- RX TX trace allignment
- GPIO Matrix Datasheet Awareness

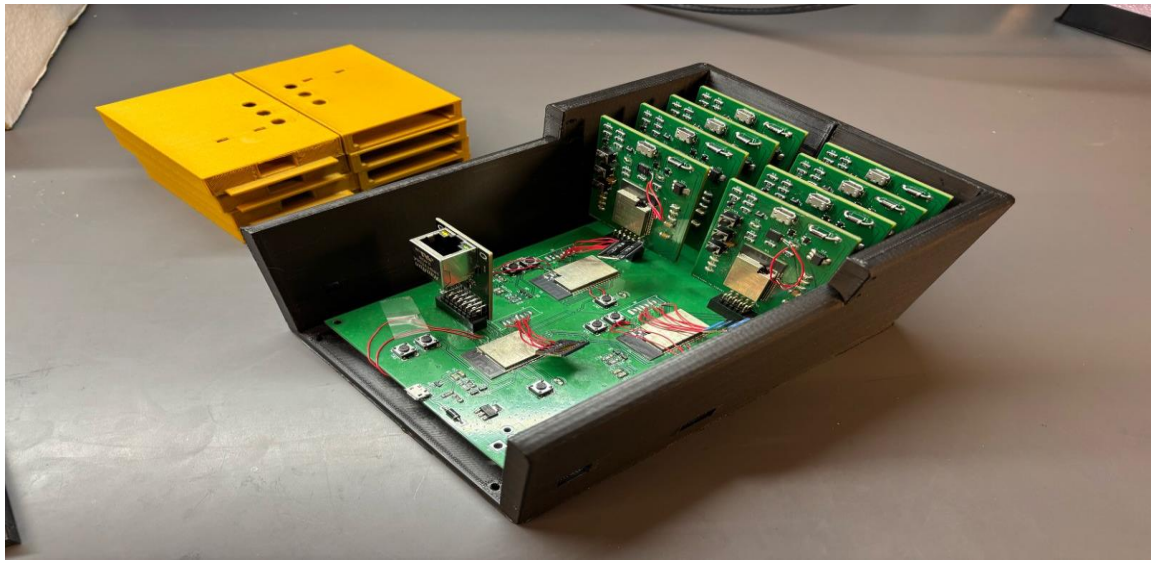
Name	No.	Type	Function
SENSOR_VP	4	I	GPIO36, ADC1_CH0, RTC_GPIO0
SENSOR_VN	5	I	GPIO39, ADC1_CH3, RTC_GPIO3
IO34	6	I	GPIO34, ADC1_CH6, RTC_GPIO4
IO35	7	I	GPIO35, ADC1_CH7, RTC_GPIO5
IO32	8	I/O	GPIO32, XTAL_32K_P (32.768 kHz crystal oscillator input), ADC1_CH4, TOUCH9, RTC_GPIO9
IO33	9	I/O	GPIO33, XTAL_32K_N (32.768 kHz crystal oscillator output), ADC1_CH5, TOUCH8, RTC_GPIO8
IO25	10	I/O	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0
IO26	11	I/O	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1



PURDUE UNIVERSITY

Project Progression!





PURDUE UNIVERSITY

Thank You

Q&A

PURDUE UNIVERSITY®