perfSONAR

Campus Perspective from University of Michigan

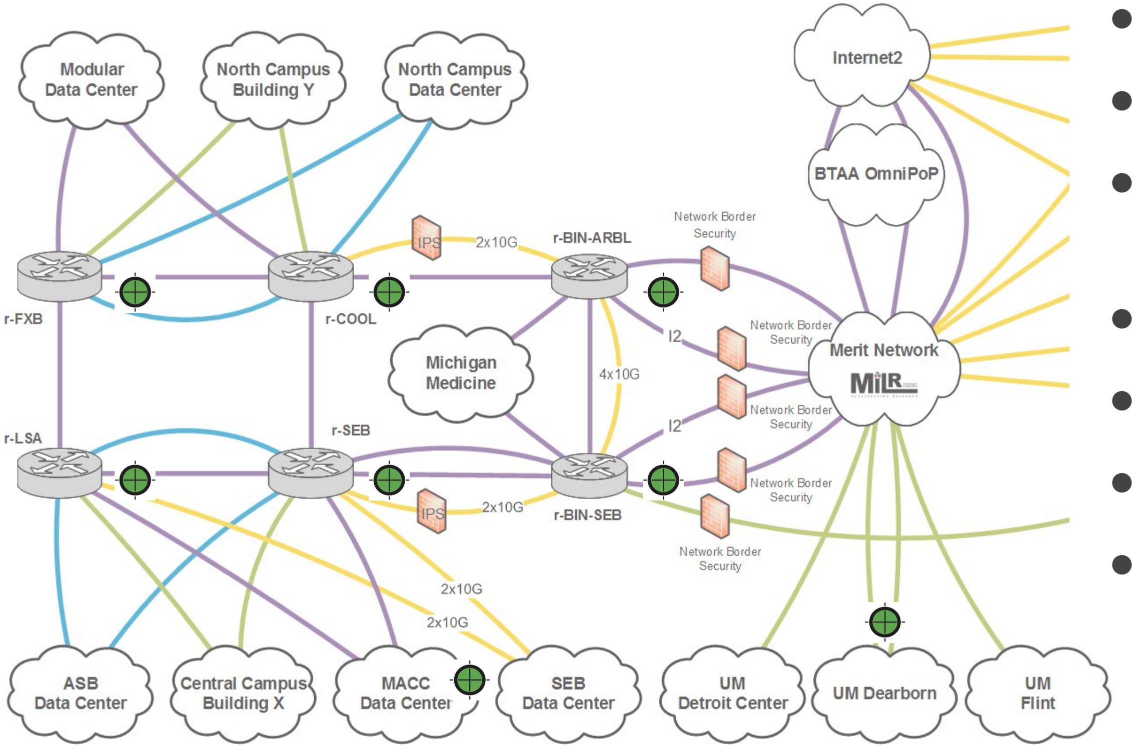
pS Automation, pS Mobile Nodes, ps Plugin Development, pS WiFi Monitoring

Edward Colone · University of Michigan · <epcjr@umich.edu>

May 24, 2022
3rd. European perfSONAR Worksop

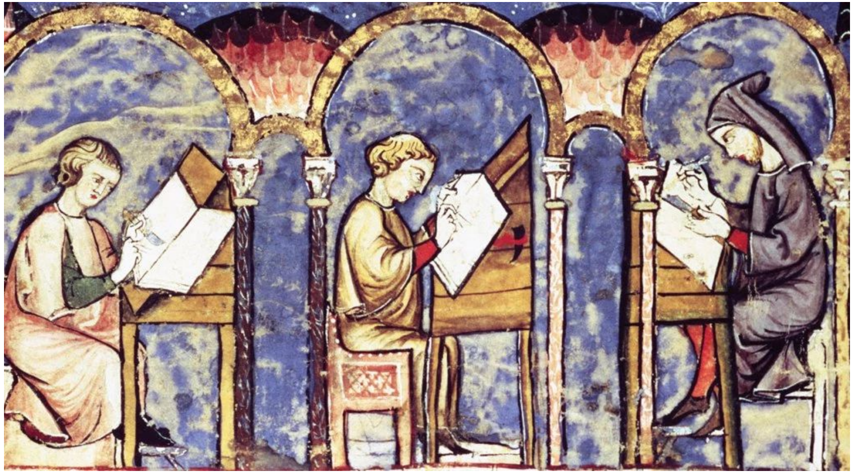
perfSONAR is developed by a partnership of

U-M perfSONAR Infrastructure



- 8 core perfSONAR Nodes
- 2 satellite campus testpoints
- Multiple 1GE and 10GE mobile diagnostic testpoints
- Esmond Data Archive
- MadDash Dashboard
- pSconfig schedule publisher
- perfSONAR Web Admin

Pre-Ansible perfSONAR Provisioning



- Major manual upgrade of a 12 node cluster: perfSONAR 3.5 / CentOS 6 → perfSONAR 4.0 / CentOS 7
  - Five UNIX sysadmins
  - Two days to complete

- Version control done manually - environment skewed
- Hard to coordinate system patching and software upgrade responsibilities between groups
- Staff time cost for node addition scaled linearly with each node added

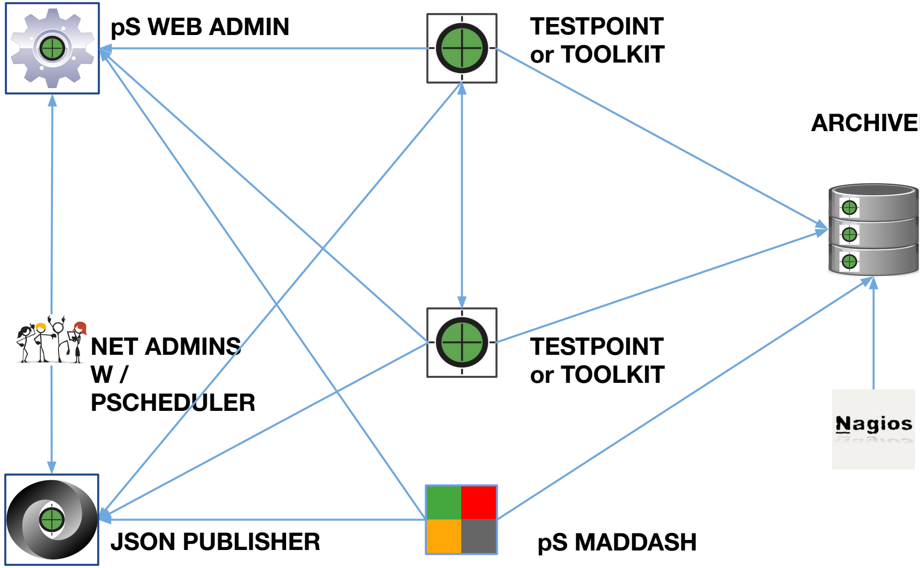
Ansible for perfSONAR

- Open source, industry standard solution
- perfSONAR authors and supports for component provisioning Ansible:
    - Master Playbook
    - Roles
- End users can bootstrap machines with base OSes, security, user accounts
- Can manage perfSONAR component interdependencies
- Config files expose perfSONAR component options / config
- Agentless, uses SSH - no extra security overhead

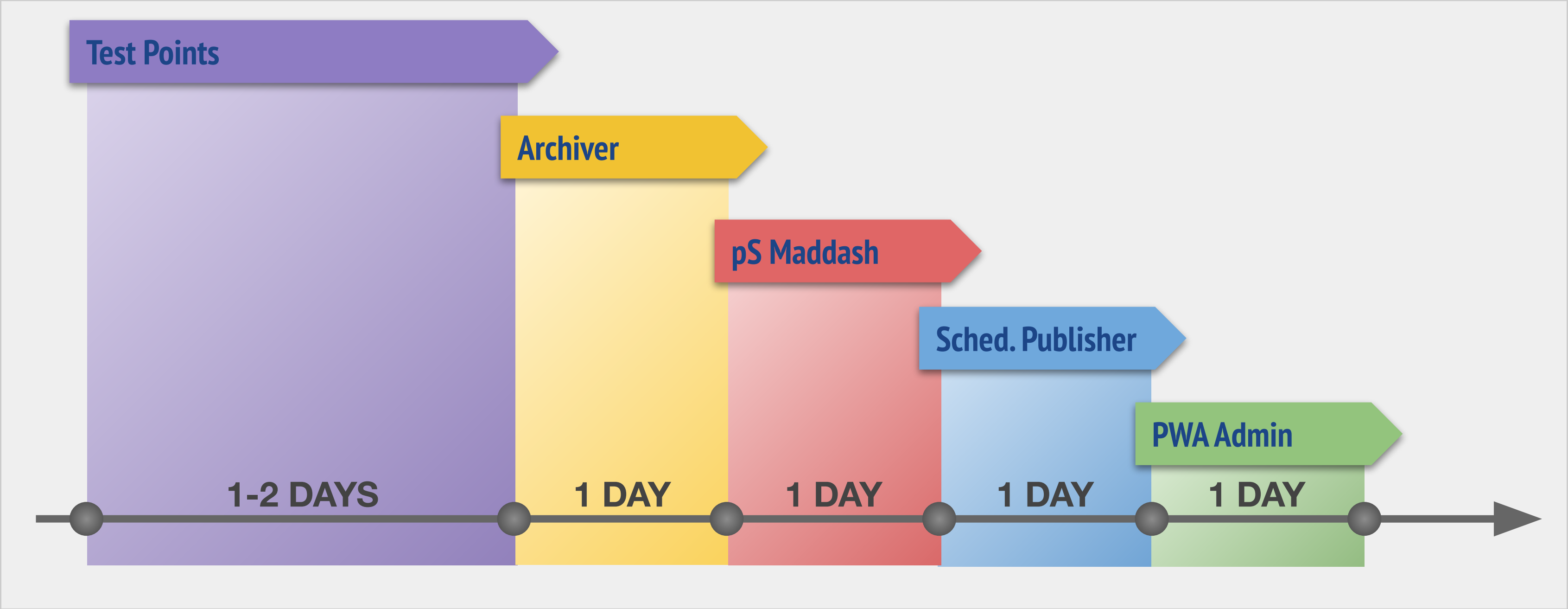
INVENTORY
ROLES
PLAYBOOK

perfSONAR: Provisioning Components

1. Archivers
2. MadDash / Dashboards
3. Testpoints
4. Toolkits
5. pSconfig raw JSON publishers
6. pSconfig Web Admin

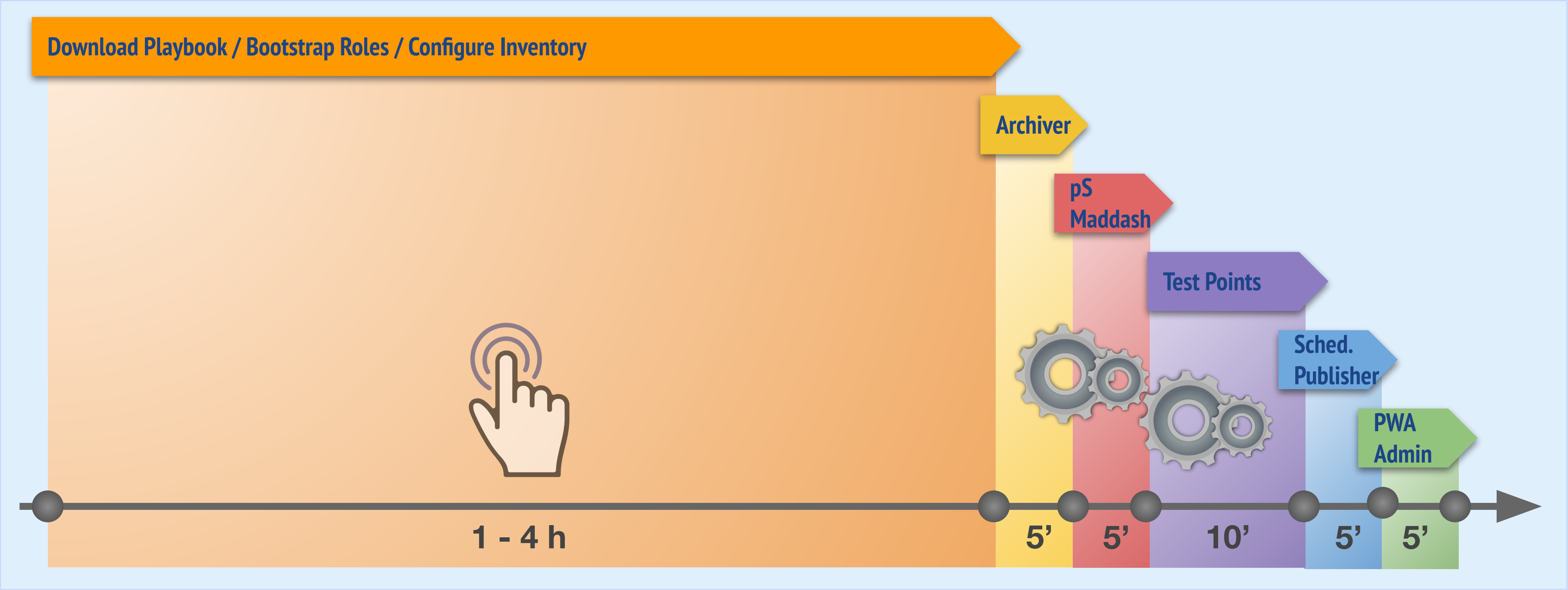


perfSONAR Manual Deployment Duration



Intermediate UNIX admin / Novice perfSONAR administrator / Full Configuration / 2-6 Testpoints

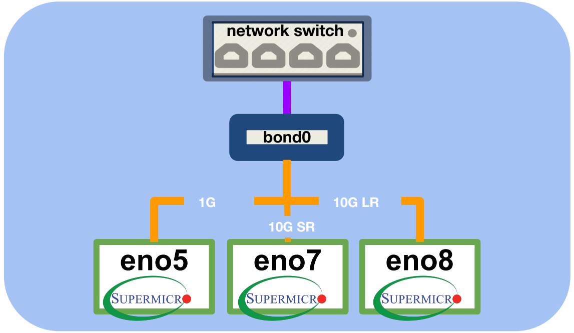
perfSONAR Ansible Deployment Duration



Intermediate UNIX admin / Novice perfSONAR administrator / Full Configuration / 2-6 Testpoints

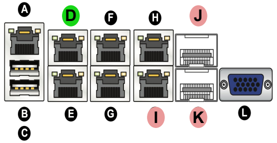
pS Mobile

Network Interface Bonding



Supermicro E300-8D

A	1GE OOB Management
D	DHCP for lab provisioning
I	BONDED 1GE
J	BONDED 10GE SR
K	BONDED 10GE LR



Rear Panel I/O		
A. IPMI LAN	E. LAN Port 1	I. LAN Port 5
B. USB Port 1	F. LAN Port 4	J. LAN Port 8 (SFP+)
C. USB Port 0	G. LAN Port 3	K. LAN Port 7 (SFP+)
D. LAN Port 2	H. LAN Port 6	L. VGA Port

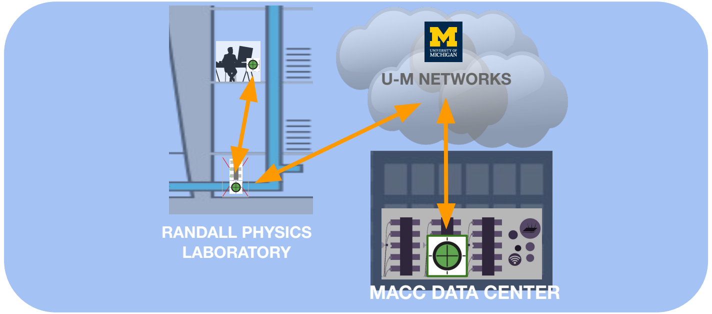
Supermicro E300-8D

Subnet Config and Test Procedure

/29 Subnet IP Usage	
Network	192.168.0.0/29
Gateway	192.168.0.1
DL1	192.168.0.2
DL2	192.168.0.3
Testpoint A	192.168.0.4
Testpoint B	192.168.0.5
Unused	192.168.0.6
Broadcast	192.168.0.7

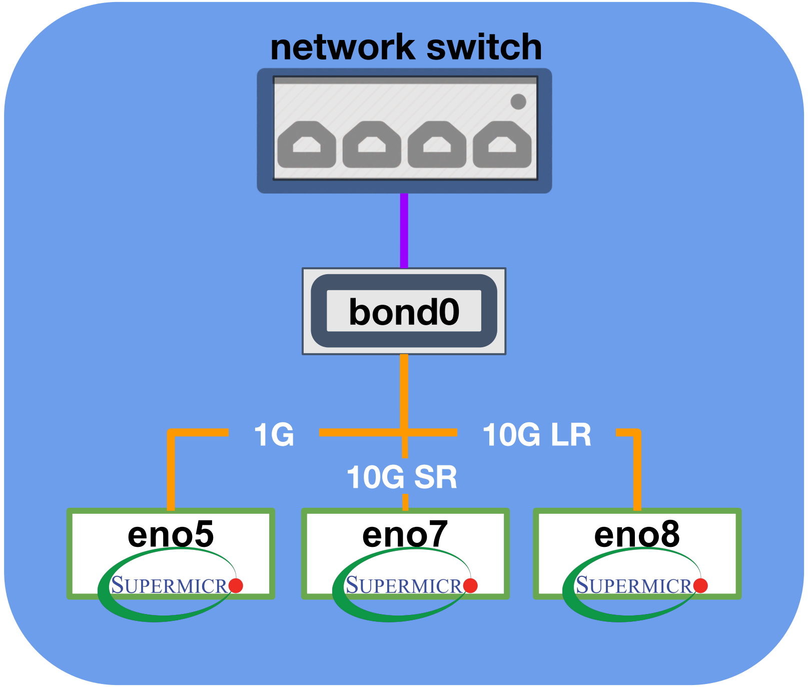
1. Build VLAN network @ test site
2. Stretch VLAN to ports you're testing through
3. Plug in the equipment, power up test hardware
4. Ping device from network
5. Log on to trusted pScheduler host
6. Run your test!

Initial Field Deployment



Network Interface Bonding

- Each interface can share a single network configuration for:
  - IP Address
  - Gateway
  - Etc. (Active/backup)
- Simplified field deployment for 1GE, 10GE SR, and 10GE LR
- Lab testing to verify adherence to performance expectations



Network Interface Bonding

A	1GE OOB Management
D	DHCP for lab provisioning
I	BONDED 1GE
J	BONDED 10GE SR
K	BONDED 10GE LR

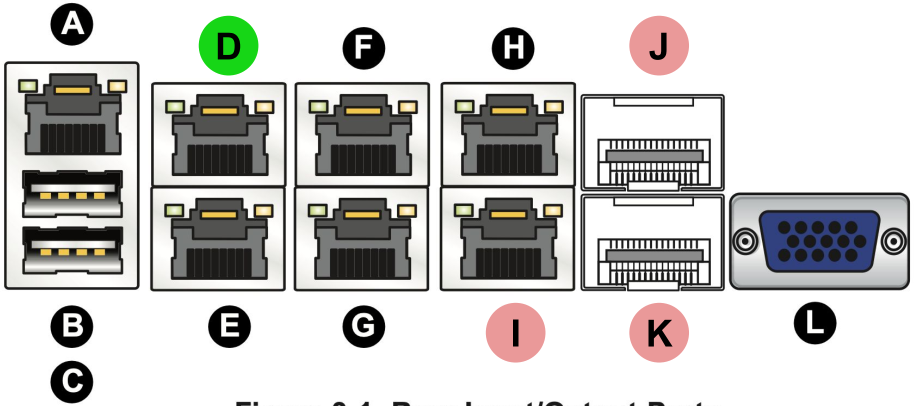
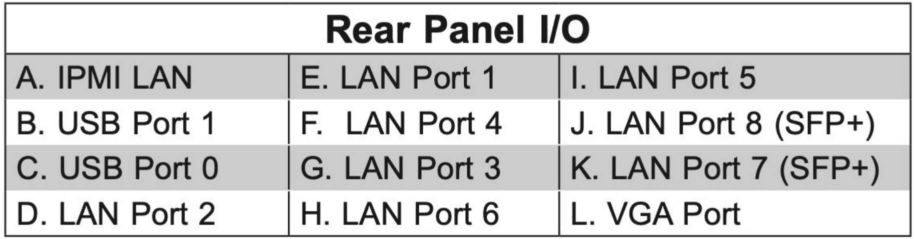


Figure 3-1. Rear Input/Output Ports



Rear Panel I/O		
A. IPMI LAN	E. LAN Port 1	I. LAN Port 5
B. USB Port 1	F. LAN Port 4	J. LAN Port 8 (SFP+)
C. USB Port 0	G. LAN Port 3	K. LAN Port 7 (SFP+)
D. LAN Port 2	H. LAN Port 6	L. VGA Port

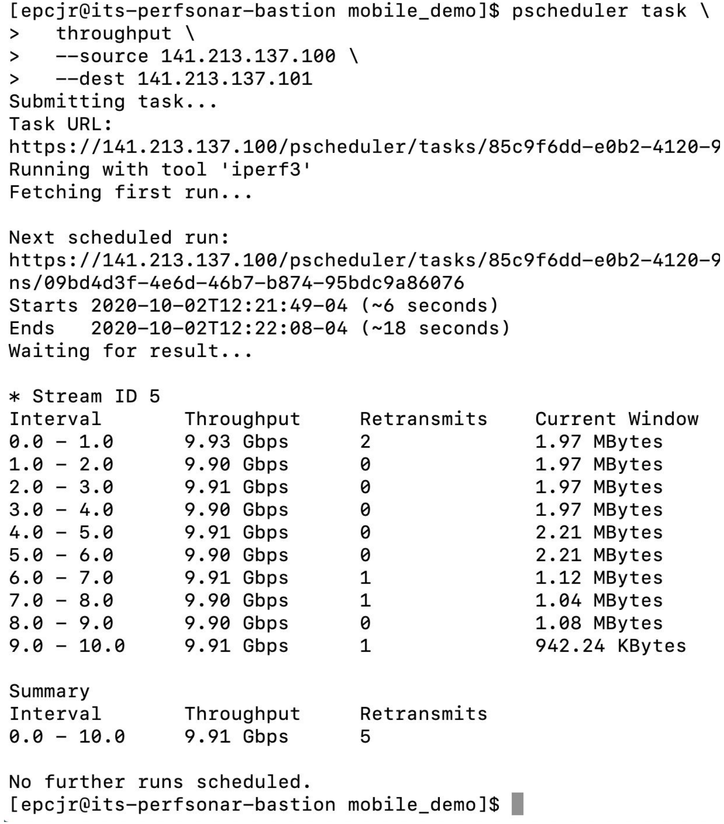
Subnet Config /29

- Dual Homed DL support
- /29 vs /30
- Troubleshooting - verify the connectivity inside VLAN
- Support for dual testpoint field deployments
- Flexibility

/29 Subnet IP Usage	
Network	192.168.0.0/29
Gateway	192.168.0.1
DL1	192.168.0.2
DL2	192.168.0.3
Testpoint A	192.168.0.4
Testpoint B	192.168.0.5
Unused	192.168.0.6
Broadcast	192.168.0.7

Test Procedure: Bastion Host & Testpoint

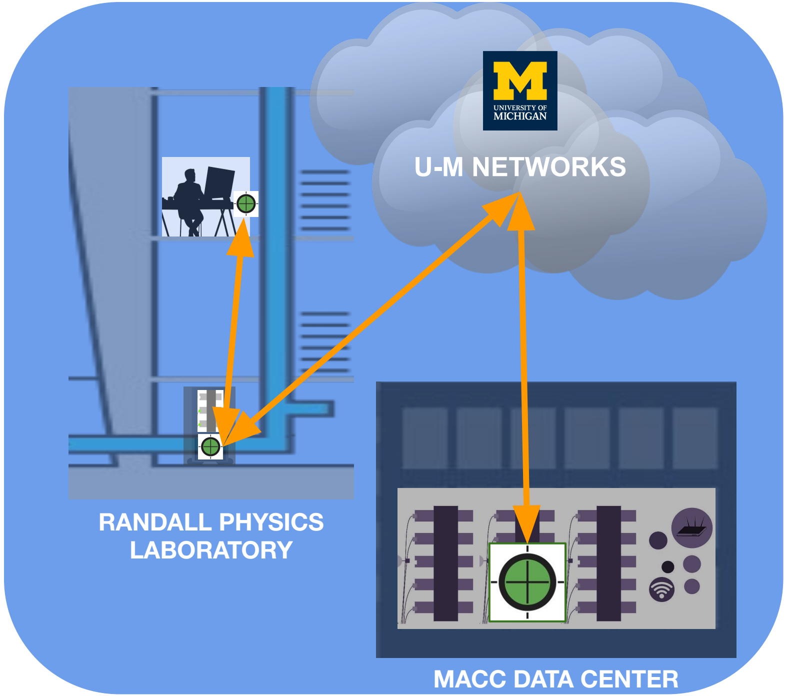
- Build VLAN network @ test site
- Stretch VLAN to ports you’re testing through
- Plug in the equipment, power up test hardware
- Ping device from network
- Log on to trusted pScheduler host
- Run your test!



```
[epcjr@its-perfsonar-bastion mobile_demo]$ pscheduler task \
>   throughput \
>   --source 141.213.137.100 \
>   --dest 141.213.137.101
Submitting task...
Task URL:
https://141.213.137.100/pscheduler/tasks/85c9f6dd-e0b2-4120-9
Running with tool 'iperf3'
Fetching first run...

Next scheduled run:
https://141.213.137.100/pscheduler/tasks/85c9f6dd-e0b2-4120-9
ns/09bd4d3f-4e6d-46b7-b874-95bdc9a86076
Starts 2020-10-02T12:21:49-04 (~6 seconds)
Ends   2020-10-02T12:22:08-04 (~18 seconds)
Waiting for result...

* Stream ID 5
Interval       Throughput     Retransmits    Current Window
0.0 - 1.0      9.93 Gbps      2              1.97 MBytes
1.0 - 2.0      9.90 Gbps      0              1.97 MBytes
2.0 - 3.0      9.91 Gbps      0              1.97 MBytes
3.0 - 4.0      9.90 Gbps      0              1.97 MBytes
4.0 - 5.0      9.91 Gbps      0              2.21 MBytes
5.0 - 6.0      9.90 Gbps      0              2.21 MBytes
6.0 - 7.0      9.91 Gbps      1              1.12 MBytes
7.0 - 8.0      9.90 Gbps      1              1.04 MBytes
8.0 - 9.0      9.90 Gbps      0              1.08 MBytes
9.0 - 10.0     9.91 Gbps      1              942.24 KBytes

Summary
Interval       Throughput     Retransmits
0.0 - 10.0     9.91 Gbps      5

No further runs scheduled.
[epcjr@its-perfsonar-bastion mobile_demo]$ 
```

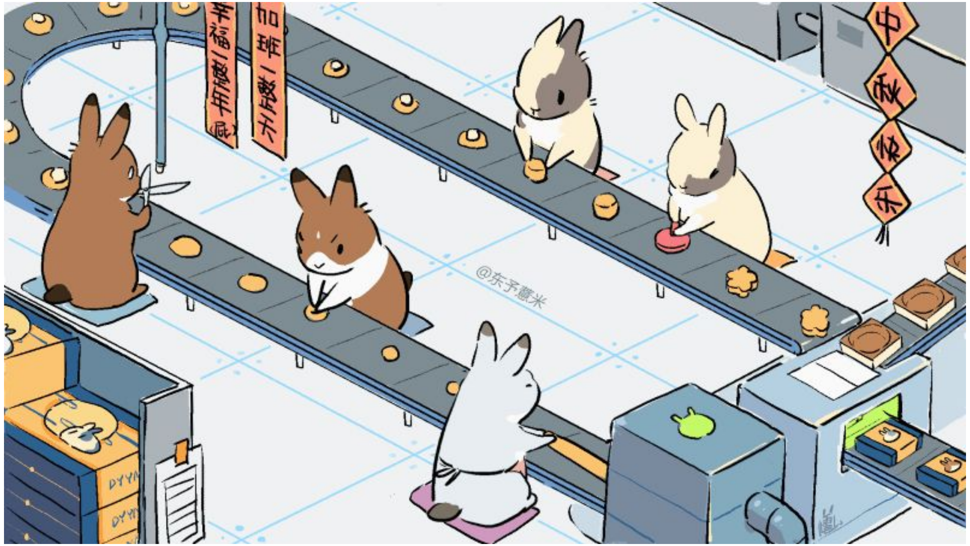
Initial Field Deployment:
End-to-End Testing - Verify Problem Exists

- Testpoint A using same fiber cabling as user
- Testpoint B in same data center rack as server
- Tests ran from user test to various permanent, temporary perfSONAR boxes
- All had issues!



I Know What You (mfeit) Did Last Summer

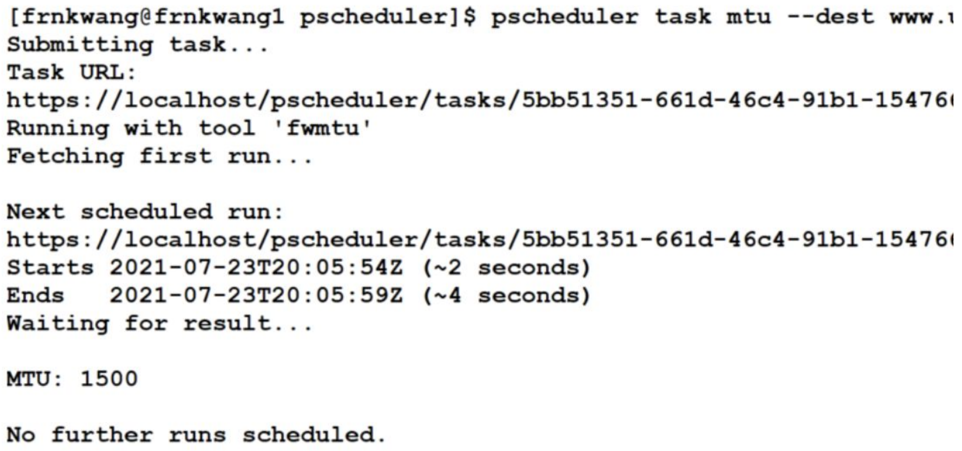
- pScheduler Plugin Development Kit
- pscheduler/scripts/PDK
- Simplify creation of tests / tools / archivers
- Creates file framework & minimum viable application
- Vagrant SDE
  - Multi OS
  - Two-participant testing



MTU Test/Tool

Test: mtu
Tool: fwmtu

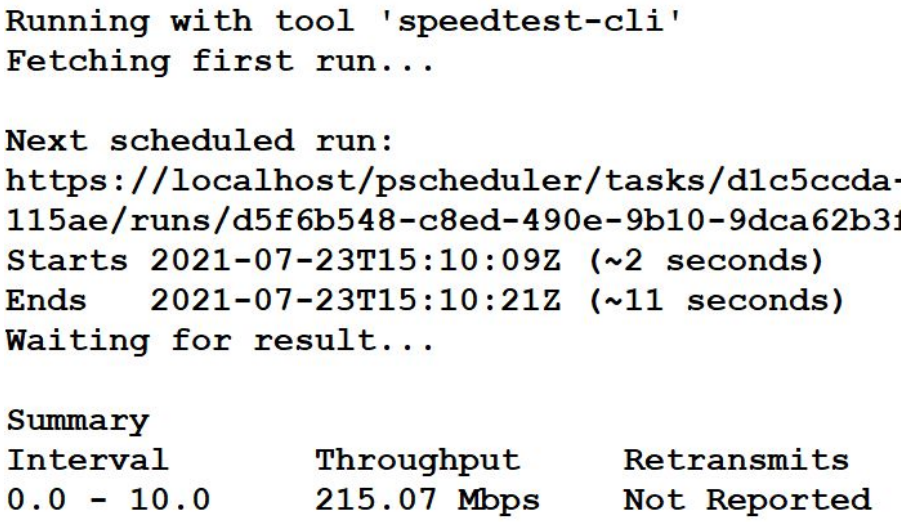
- Uses: Determine path MTU to a destination
- Usage case: Troubleshooting problems for if packet size is too large
  - Ex: if control packets are small, they can get through, but data packets might not
- CL Args:
  - --dest: destination (required)
  - --port: optional port (defaults to 1060)
- How it works: sends out a large packet and uses Linux's “Discover MTU” option



```
[frnkwang@frnkwang1 pscheduler]$ pscheduler task mtu --dest www.
Submitting task...
Task URL:
https://localhost/pscheduler/tasks/5bb51351-661d-46c4-91b1-15476
Running with tool 'fwmtu'
Fetching first run...

Next scheduled run:
https://localhost/pscheduler/tasks/5bb51351-661d-46c4-91b1-15476
Starts 2021-07-23T20:05:54Z (~2 seconds)
Ends   2021-07-23T20:05:59Z (~4 seconds)
Waiting for result...

MTU: 1500

No further runs scheduled.
```

Speedtest-CLI Tool

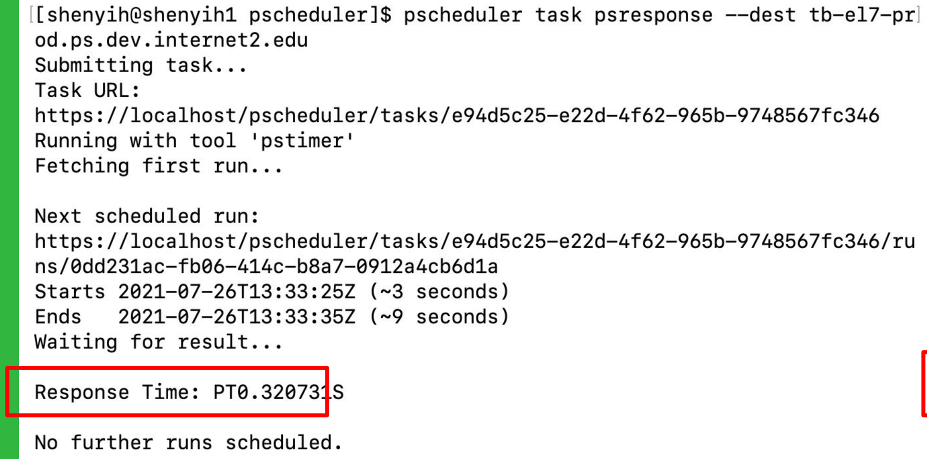
- Uses the open-source implementation of Ookla’s Speedtest Command Line Interface.
- Under Throughput Test
- Measures the throughput of the network.



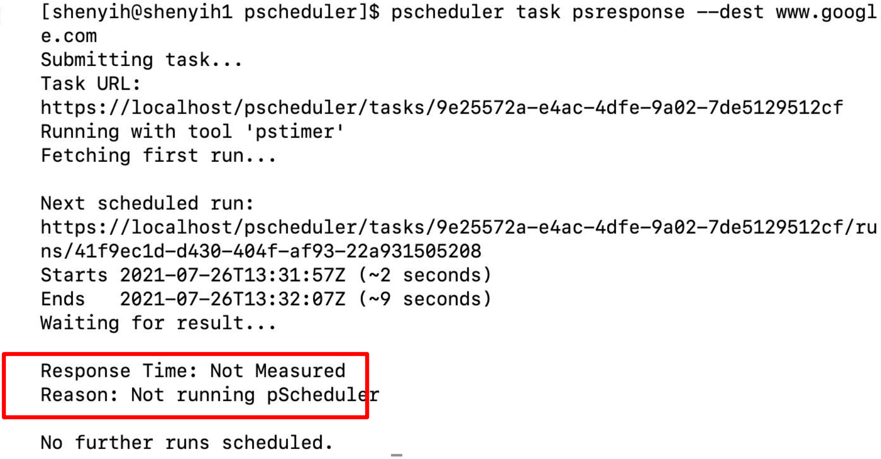
```
Running with tool 'speedtest-cli'
Fetching first run...

Next scheduled run:
https://localhost/pscheduler/tasks/d1c5ccda-
115ae/runs/d5f6b548-c8ed-490e-9b10-9dca62b3f
Starts 2021-07-23T15:10:09Z (~2 seconds)
Ends   2021-07-23T15:10:21Z (~11 seconds)
Waiting for result...

Summary
Interval        Throughput      Retransmits
0.0 - 10.0      215.07 Mbps     Not Reported
```

pScheduler psresponse Test

- Primary use: checks pScheduler nodes are functioning and how long they take to respond
- Command Line Arguments
    - --dest is required



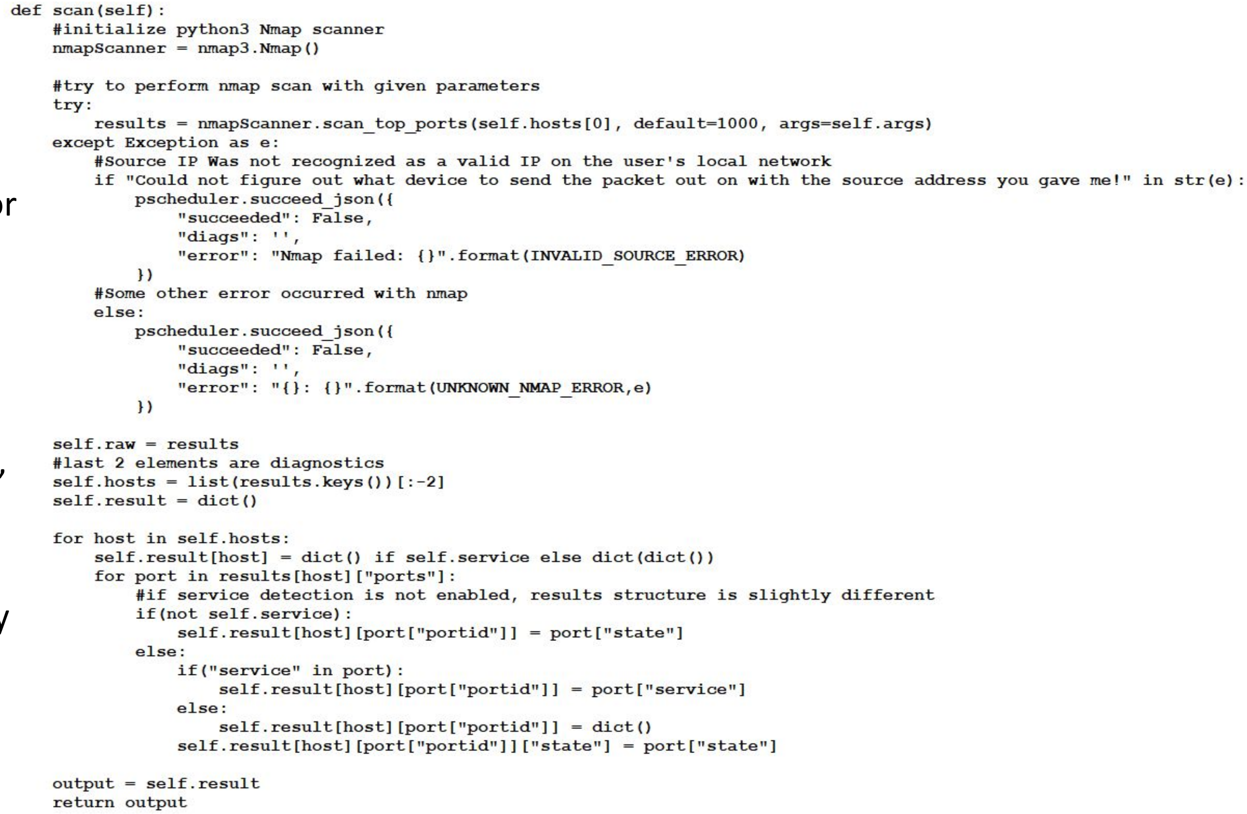
```
[shenyih@shenyih1 pscheduler]$ pscheduler task psresponse --dest tb-el7-pr
od.ps.dev.internet2.edu
Submitting task...
Task URL:
https://localhost/pscheduler/tasks/e94d5c25-e22d-4f62-965b-9748567fc346
Running with tool 'pstimer'
Fetching first run...

Next scheduled run:
https://localhost/pscheduler/tasks/e94d5c25-e22d-4f62-965b-9748567fc346/ru
ns/0dd231ac-fb06-414c-b8a7-0912a4cb6d1a
Starts 2021-07-26T13:33:25Z (~3 seconds)
Ends   2021-07-26T13:33:35Z (~9 seconds)
Waiting for result...

Response Time: PT0.320731S

No further runs scheduled.
```



```
[shenyih@shenyih1 pscheduler]$ pscheduler task psresponse --dest www.googl
e.com
Submitting task...
Task URL:
https://localhost/pscheduler/tasks/9e25572a-e4ac-4dfe-9a02-7de5129512cf
Running with tool 'pstimer'
Fetching first run...

Next scheduled run:
https://localhost/pscheduler/tasks/9e25572a-e4ac-4dfe-9a02-7de5129512cf/ru
ns/41f9ec1d-d430-404f-af93-22a931505208
Starts 2021-07-26T13:31:57Z (~2 seconds)
Ends   2021-07-26T13:32:07Z (~9 seconds)
Waiting for result...

Response Time: Not Measured
Reason: Not running pScheduler

No further runs scheduled.
```

OpenPorts

- Test: openports
- Tool: nmapscan
- Scan a network or subnet to check for open and filtered ports
- Options: Service Detection, Specific Port Range, Source IP Specification
    - Service Detection: Can display state, product name, version, OS, and other information for each port
- Useful for auditing the security policy in place on a given network and identifying vulnerabilities

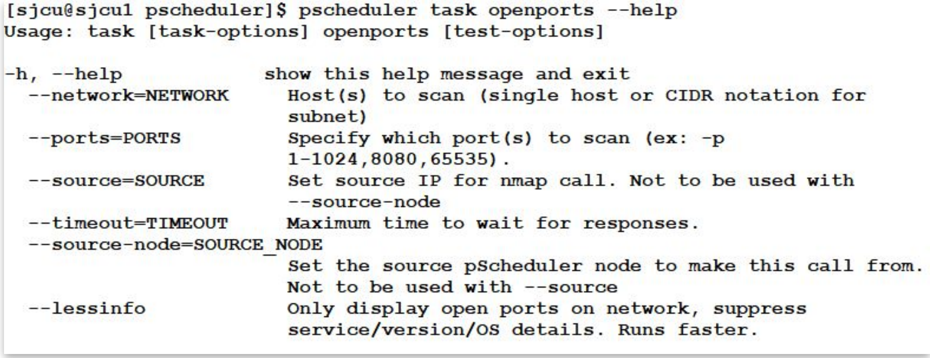


```
def scan(self):
    #initialize python3 Nmap scanner
    nmapScanner = nmap3.Nmap()

    #try to perform nmap scan with given parameters
    try:
        results = nmapScanner.scan_top_ports(self.hosts[0], default=1000, args=self.args)
    except Exception as e:
        #Source IP Was not recognized as a valid IP on the user's local network
        if "Could not figure out what device to send the packet out on with the source address you gave me!" in str(e):
            pscheduler.succeed_json({
                "succeeded": False,
                "diags": '',
                "error": "Nmap failed: {}".format(INVALID_SOURCE_ERROR)
            })
        #Some other error occurred with nmap
        else:
            pscheduler.succeed_json({
                "succeeded": False,
                "diags": '',
                "error": "{}: {}".format(UNKNOWN_NMAP_ERROR,e)
            })

    self.raw = results
    #last 2 elements are diagnostics
    self.hosts = list(results.keys())[:-2]
    self.result = dict()

    for host in self.hosts:
        self.result[host] = dict() if self.service else dict(dict())
        for port in results[host]["ports"]:
            #if service detection is not enabled, results structure is slightly different
            if(not self.service):
                self.result[host][port["portid"]] = port["state"]
            else:
                if("service" in port):
                    self.result[host][port["portid"]] = port["service"]
                else:
                    self.result[host][port["portid"]] = dict()
                self.result[host][port["portid"]]["state"] = port["state"]

    output = self.result
    return output
```

INFORMATION AND TECHNOLOGY SERVICES
UNIVERSITY OF MICHIGAN

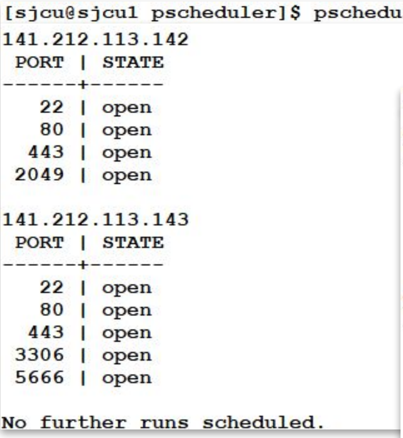
OpenPorts

Help message



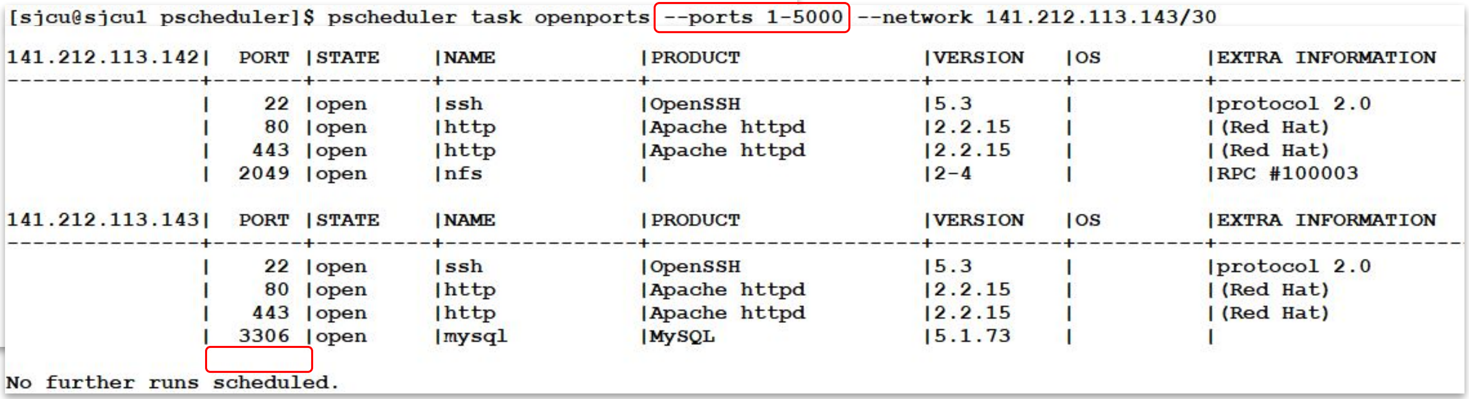
```
[sjcu@sjcu1 pscheduler]$ pscheduler task openports --help
Usage: task [task-options] openports [test-options]

-h, --help            show this help message and exit
  --network=NETWORK     Host(s) to scan (single host or CIDR notation for
                        subnet)
  --ports=PORTS         Specify which port(s) to scan (ex: -p
                        1-1024,8080,65535).
  --source=SOURCE       Set source IP for nmap call. Not to be used with
                        --source-node
  --timeout=TIMEOUT     Maximum time to wait for responses.
  --source-node=SOURCE_NODE
                        Set the source pScheduler node to make this call from.
                        Not to be used with --source
  --lessinfo            Only display open ports on network, suppress
                        service/version/OS details. Runs faster.
```

Reduced Format



```
[sjcu@sjcu1 pscheduler]$ pscheduler task openports --lessinfo --network 141.212.113.143/30
141.212.113.142
 PORT | STATE
------+------
   22 | open
   80 | open
  443 | open
 2049 | open

141.212.113.143
 PORT | STATE
------+------
   22 | open
   80 | open
  443 | open
 3306 | open
 5666 | open

No further runs scheduled.
```

Standard Format



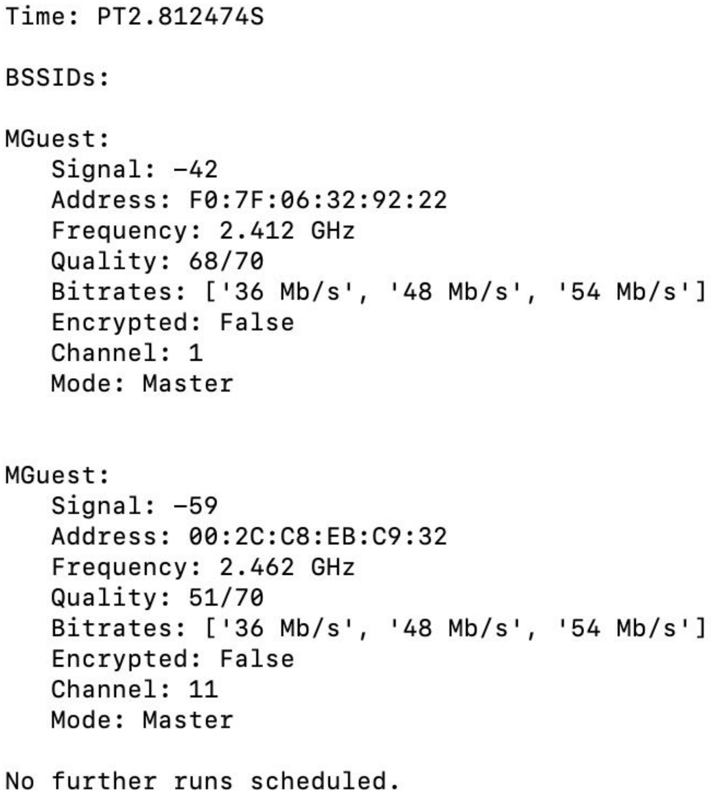
```
[sjcu@sjcu1 pscheduler]$ pscheduler task openports --ports 1-5000 --network 141.212.113.143/30
141.212.113.142|  PORT |STATE    |NAME            |PRODUCT           |VERSION   |OS         |EXTRA INFORMATION
---------------+-------+---------+----------------+------------------+----------+-----------+------------------
               |    22 |open     |ssh             |OpenSSH           |5.3       |           |protocol 2.0
               |    80 |open     |http            |Apache httpd      |2.2.15    |           |(Red Hat)
               |   443 |open     |http            |Apache httpd      |2.2.15    |           |(Red Hat)
               |  2049 |open     |nfs             |                  |2-4       |           |RPC #100003

141.212.113.143|  PORT |STATE    |NAME            |PRODUCT           |VERSION   |OS         |EXTRA INFORMATION
---------------+-------+---------+----------------+------------------+----------+-----------+------------------
               |    22 |open     |ssh             |OpenSSH           |5.3       |           |protocol 2.0
               |    80 |open     |http            |Apache httpd      |2.2.15    |           |(Red Hat)
               |   443 |open     |http            |Apache httpd      |2.2.15    |           |(Red Hat)
               |  3306 |open     |mysql           |MySQL             |5.1.73    |           |

No further runs scheduled.
```

BSSID Scan

Test: wifibssid

Tool: bssidscanner

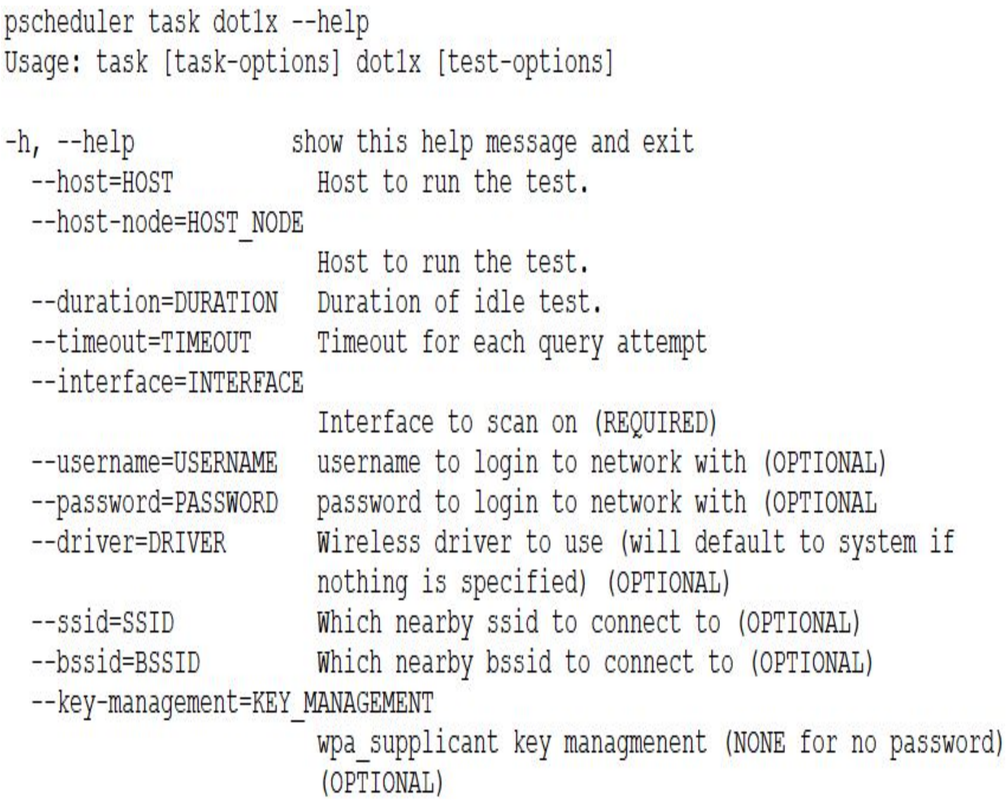
- **Use:** Returns a list of all associated BSSIDs in json format for the given SSID
- **Usage case:** Confirm that all access points for a SSID are working properly
- **Command Line Arguments**
  - --interface: interface that will be scanned for BSSIDs (required)
  - --ssid: List of BSSIDs returned will be associated with this ssid



```
Time: PT2.812474S

BSSIDs:

MGuest:
   Signal: -42
   Address: F0:7F:06:32:92:22
   Frequency: 2.412 GHz
   Quality: 68/70
   Bitrates: ['36 Mb/s', '48 Mb/s', '54 Mb/s']
   Encrypted: False
   Channel: 1
   Mode: Master


MGuest:
   Signal: -59
   Address: 00:2C:C8:EB:C9:32
   Frequency: 2.462 GHz
   Quality: 51/70
   Bitrates: ['36 Mb/s', '48 Mb/s', '54 Mb/s']
   Encrypted: False
   Channel: 11
   Mode: Master

No further runs scheduled.
```

dot1x Test

- **Use**: Authenticate to a wifi network using 802.1x
- **Usage case**: When there is a machine running pScheduler that needs to connect through a wireless network
    - Ex: pScheduler node is not hooked up on ethernet but has a wireless capabilities, this plugin can authenticate it to an access point.
- **CL Args**:



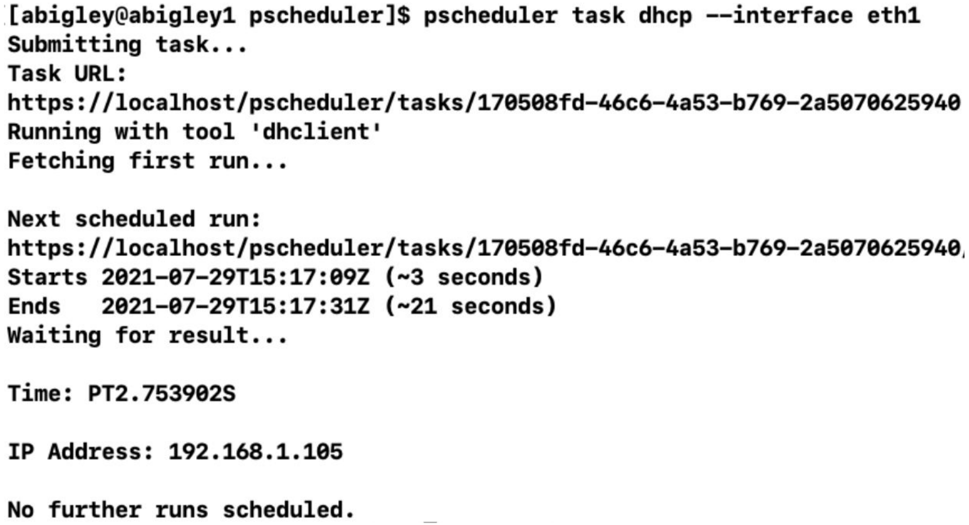
```
pscheduler task dot1x --help
Usage: task [task-options] dot1x [test-options]

-h, --help            show this help message and exit
  --host=HOST           Host to run the test.
  --host-node=HOST_NODE
                        Host to run the test.
  --duration=DURATION   Duration of idle test.
  --timeout=TIMEOUT     Timeout for each query attempt
  --interface=INTERFACE
                        Interface to scan on (REQUIRED)
  --username=USERNAME   username to login to network with (OPTIONAL)
  --password=PASSWORD   password to login to network with (OPTIONAL
  --driver=DRIVER       Wireless driver to use (will default to system if
                        nothing is specified) (OPTIONAL)
  --ssid=SSID           Which nearby ssid to connect to (OPTIONAL)
  --bssid=BSSID         Which nearby bssid to connect to (OPTIONAL)
  --key-management=KEY_MANAGEMENT
                        wpa_supplicant key managmenent (NONE for no password)
                        (OPTIONAL)
```

DHCP Response Time

Test: dhcp

Tool: dhclient

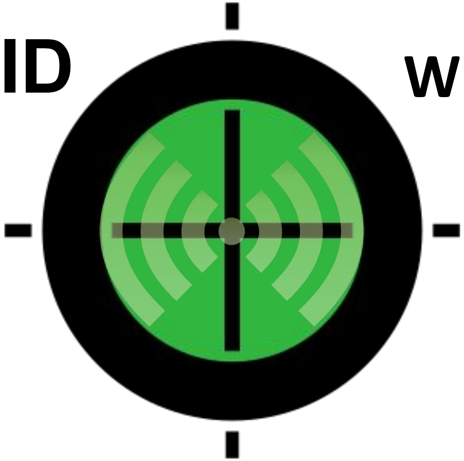
- --interface - specify a particular interface to run dhclient on
  - If not specified, dhclient uses the first interface configured on system
- Uses linux dhclient command
- Releases the current ip address on the interface
- Requests a new ip address
- Allows users to check that DHCP server performance is consistent with expectations



```
[abigley@abigley1 pscheduler]$ pscheduler task dhcp --interface eth1
Submitting task...
Task URL:
https://localhost/pscheduler/tasks/170508fd-46c6-4a53-b769-2a5070625940
Running with tool 'dhclient'
Fetching first run...

Next scheduled run:
https://localhost/pscheduler/tasks/170508fd-46c6-4a53-b769-2a5070625940
Starts 2021-07-29T15:17:09Z (~3 seconds)
Ends   2021-07-29T15:17:31Z (~21 seconds)
Waiting for result...

Time: PT2.753902S

IP Address: 192.168.1.105

No further runs scheduled.
```

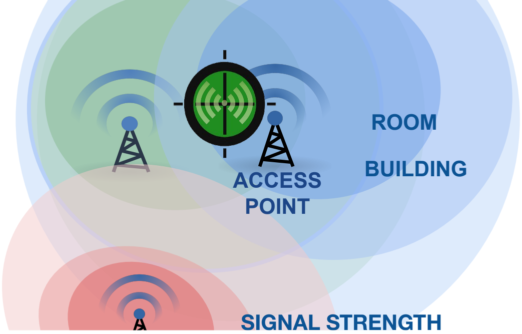
pSSID WiFi Monitor

perfSONAR

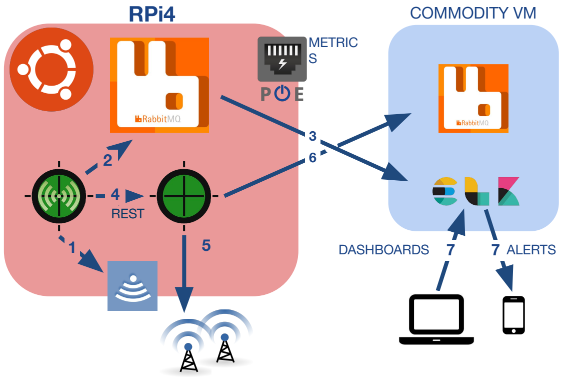


Use Case

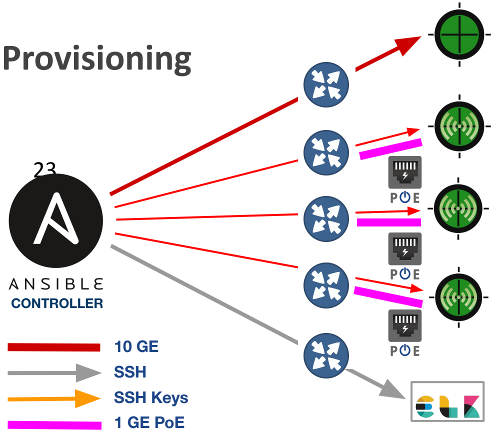
Multiple APs, Mixed Coverage, Multiple SSID/BSSID

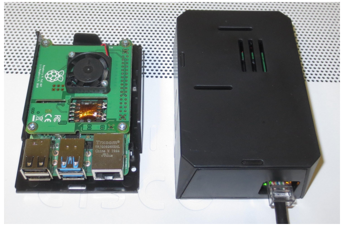


Architecture

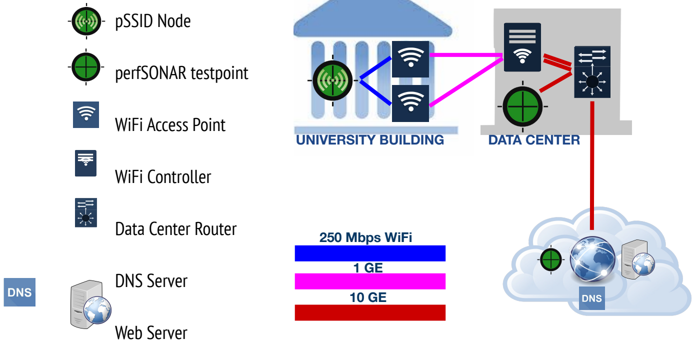


Provisioning





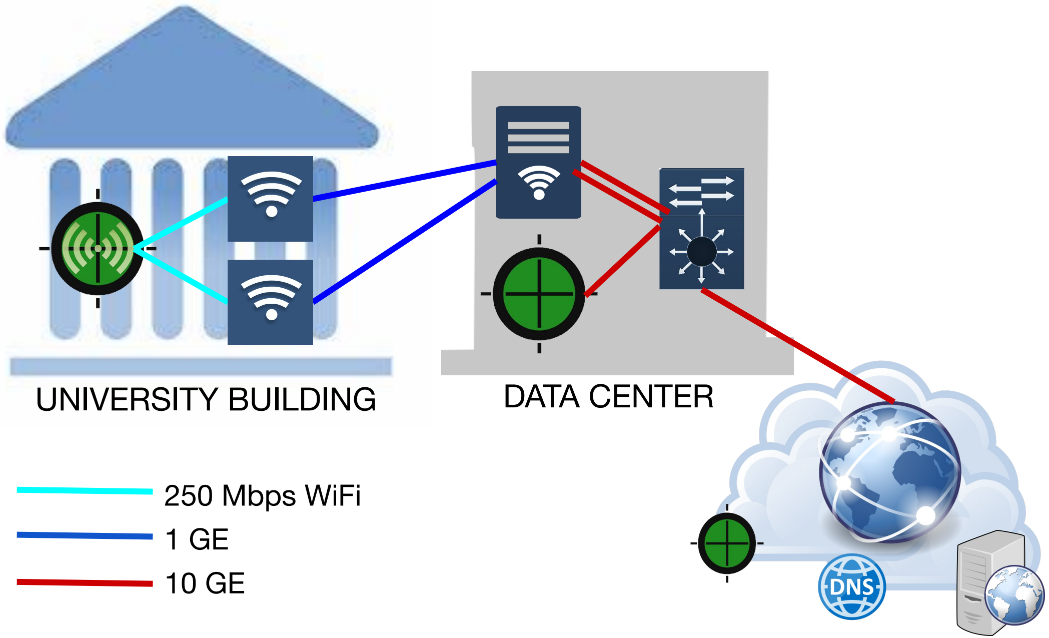
Deployment



ESnet GÉANT INDIANA UNIVERSITY INTERNET2 UNIVERSITY OF MICHIGAN

WiFi Physical Testing Architecture

- pSSID Node
- perfSONAR testpoint
- WiFi Access Point
- WiFi Controller
- Data Center Router
- DNS Server
- Web Server



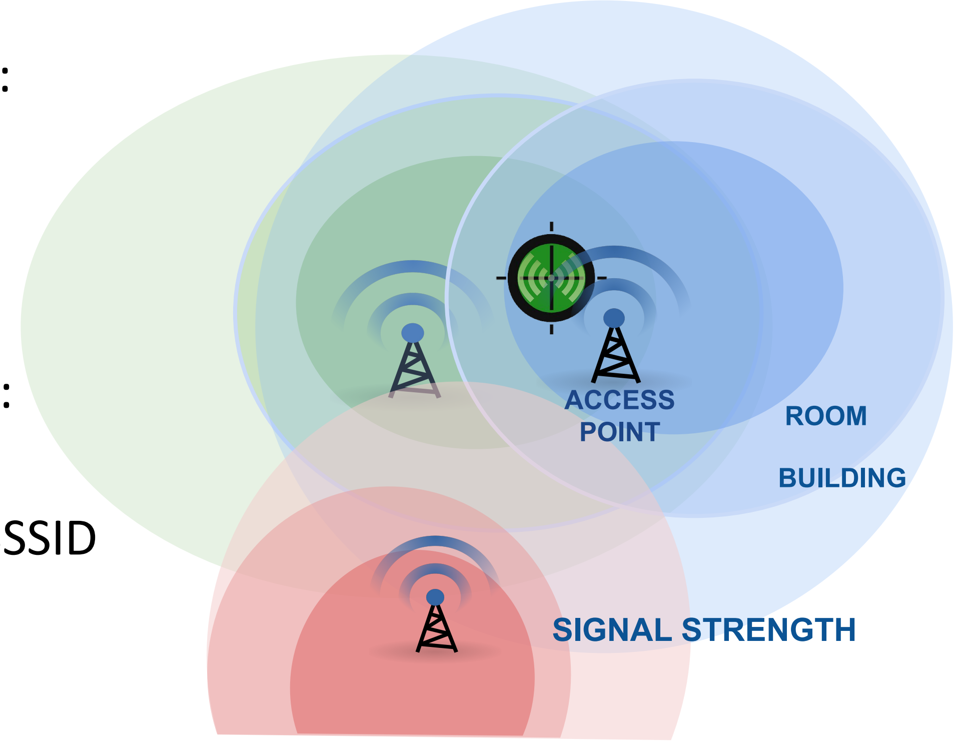
Scenario: Multiple APs, Mixed Coverage

Passive Scanning for all Access Points:

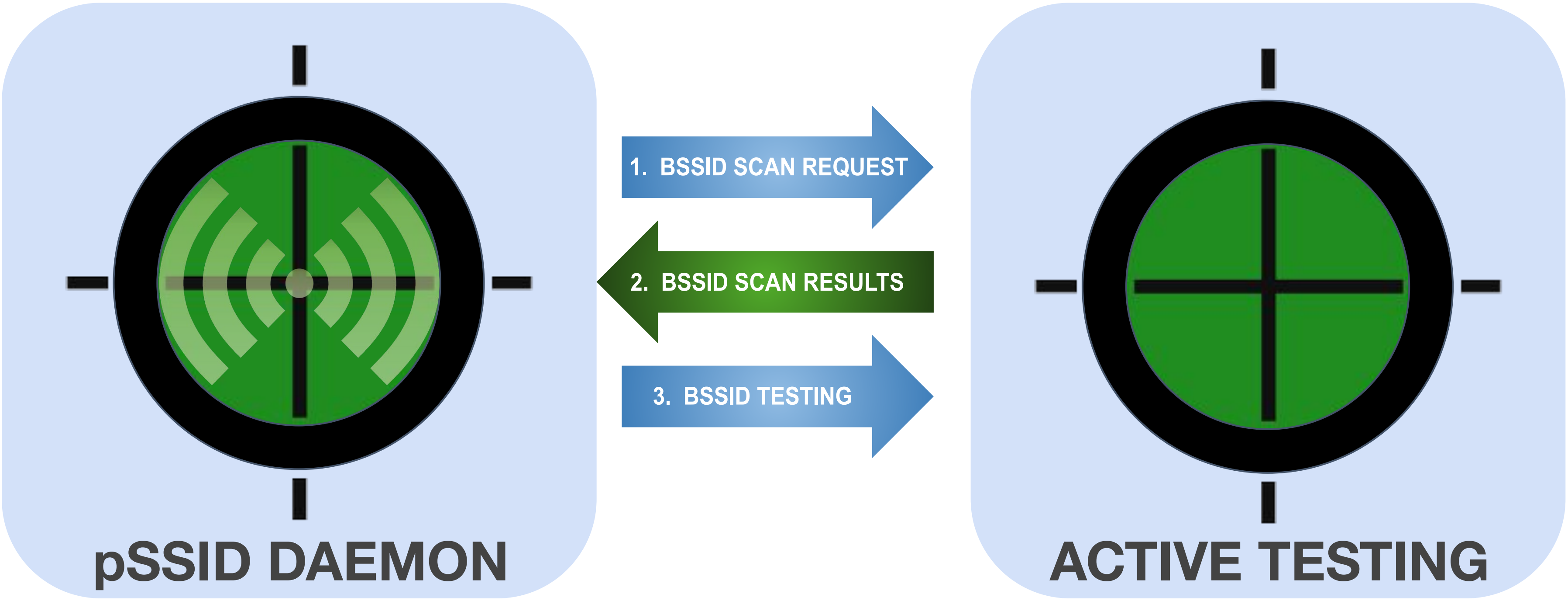
- Rogue SSIDs
- BSSID Channel Mismatch
- Insufficient Coverage

Active Testing for strong Access Point:

- Connect & Test qualifying BSSIDs
- Test results are correlated with BSSID



WiFi Testing with pPSSID

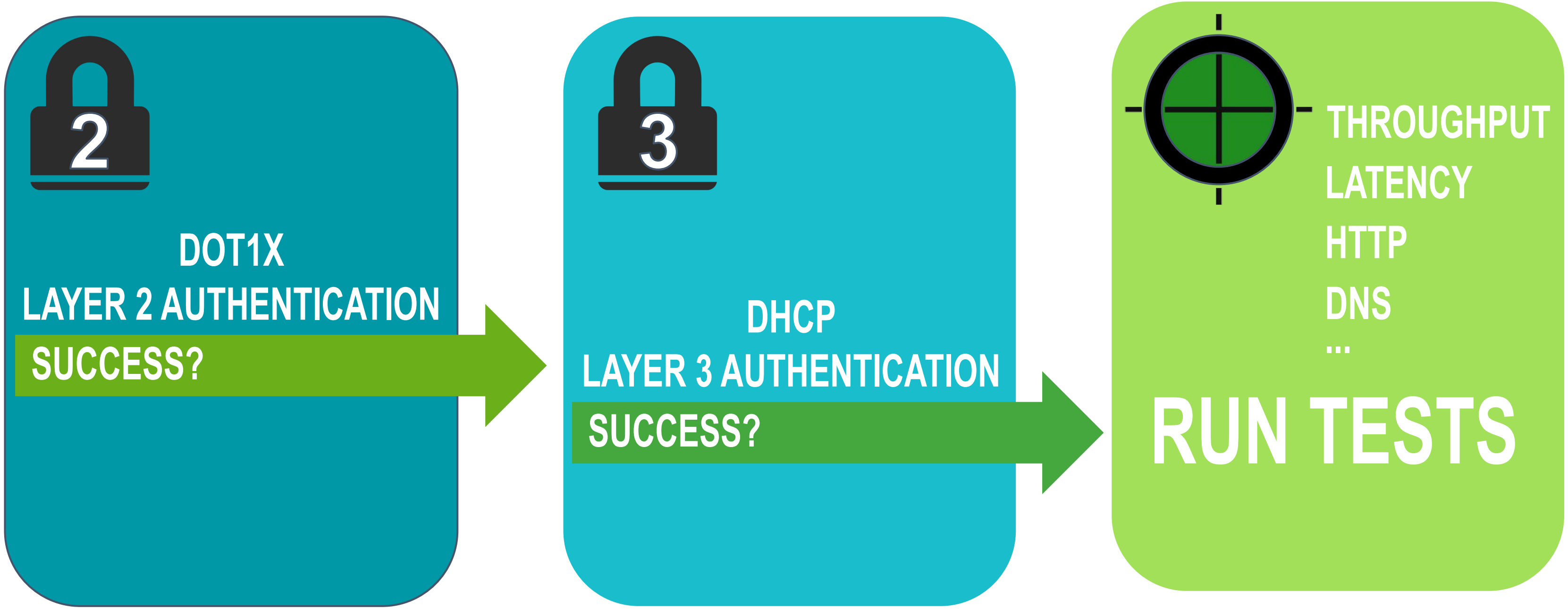


pSSID DAEMON

1. BSSID SCAN REQUEST
2. BSSID SCAN RESULTS
3. BSSID TESTING

ACTIVE TESTING

pSSID Batch Process



2

DOT1X

LAYER 2 AUTHENTICATION

SUCCESS?

3

DHCP

LAYER 3 AUTHENTICATION

SUCCESS?

THROUGHPUT

LATENCY

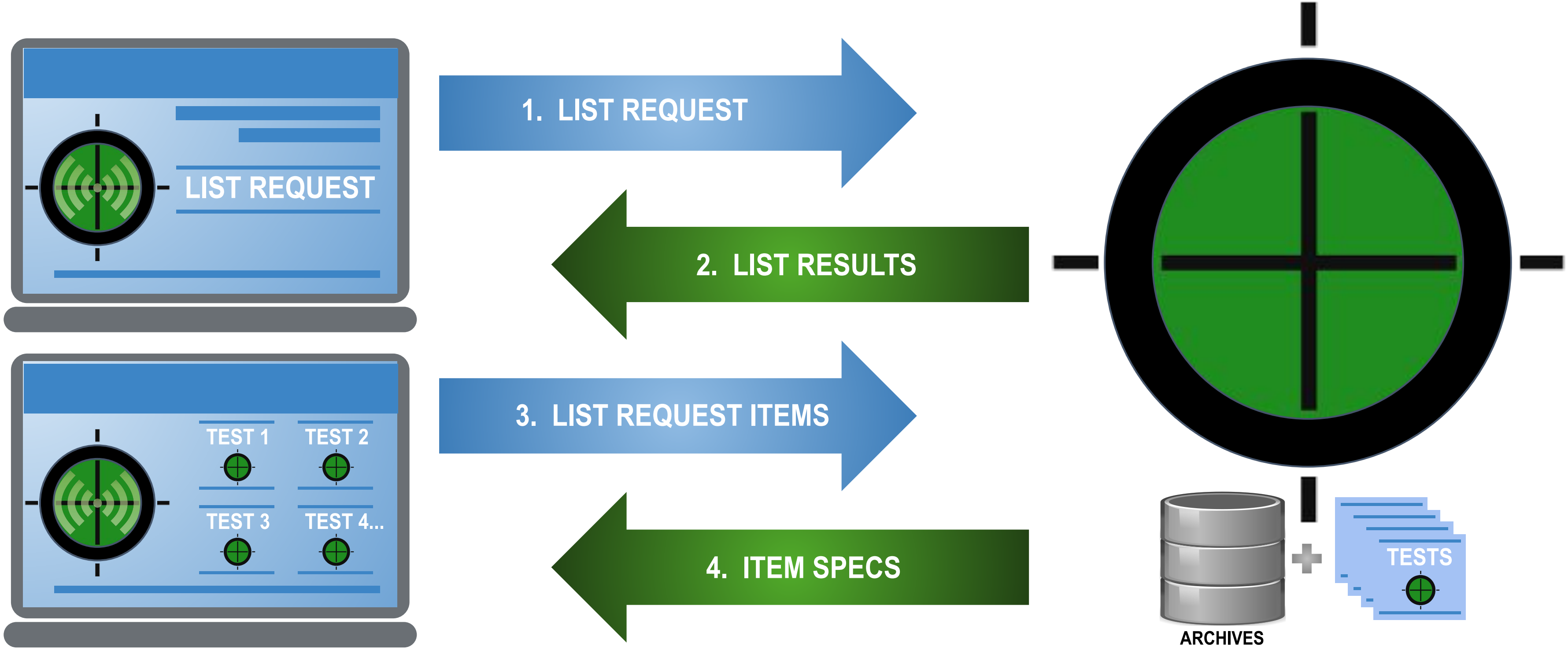
HTTP

DNS

…

RUN TESTS

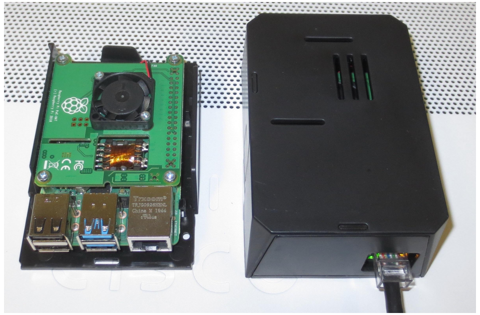
PSSID Dynamic Configuration Graphic Interface



INFORMATION AND TECHNOLOGY SERVICES
UNIVERSITY OF MICHIGAN

Trial Field Deployment

- Alpha field test / deployment
  - Off-the-shelf case with port security
  - PoE Hat
  - 64GB SD Card
  - ~$100 per complete probe
- Refine Deployment and Configuration model
  - Automated OS provisioning
  - Application provisioning via Ansible
  - Individual config files / schedules
  - Refine ELK & RMQ integration & deployments



Questions