Detection 3

Intrusion detection with SOCs Part 2

SOC deployment and operation

David Crooks

UKRI STFC

EGI CSIRT/IRIS Security team

david.crooks@stfc.ac.uk

Recap

- From yesterday, discussed logging methodologies and tools
  - rsyslog
  - OSSEC/Wazuh

- Traceability
  - Which logs are necessary and most useful

Recap

- Network logging
  - Net/sflow: usually sampled network metadata
  - Deep packet inspection nIDS: forensic level monitoring

- SOC
  - People
  - Processes
  - Technology

Recap

- Technology:
  - Threat intelligence
  - Fine-grained network monitoring
  - Storage and Visualisation
  - Alerting

Recap

- MISP: Threat Intelligence
- Zeek: Network logging
- OpenSearch: Storage & Visualisation

SOC Deployment and Operation

- Look in more detail into deploying and operating a SOC

- Use STFC as a worked example
  - Based on CERN’s experience

Architecture

- Let’s think about our architecture!

- We need to tap 6 external links
  - 2 x 100 Gb/s LHCOPN link → Science data
  - 4 x 100 Gb/s Janet links → All site data including laptops/desktops
    - 2 active at any one time

Architecture

- This cluster is going to contain sensitive traffic
  - Monitors all traffic offsite

- Need to design our network architecture and deployment plan with care

Router taps

- Recall
- Three ways to tap network traffic
    - Optical taps
    - Port mirroring
    - Packet broker

Optical taps

- Physical, passive splitters
- Splits the light from one fibre into two
- Can use 50/50, 60/40, 70/30

- Decision will be made based on length of fibre, amount of traffic, experience…

Port mirroring

- Or port spanning
- Use router/switch to mirror traffic from one port to another
- Most hardware has this capability

Packet Broker

- Dedicated network equipment
- Increased flexibility and capability
- Expensive
- Consider when monitoring multiple 100Gb/s links

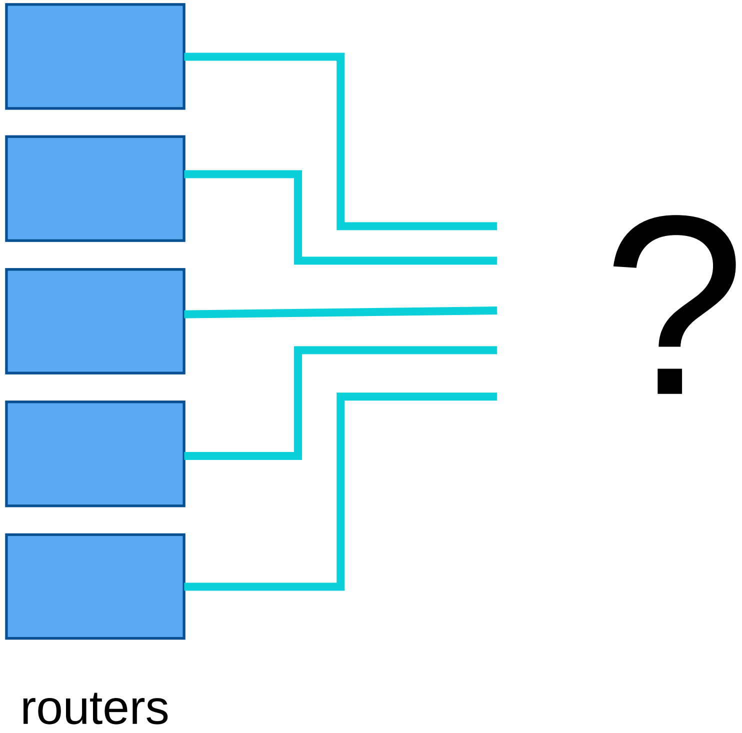
Pros and cons

	Pros	Cons
Optical taps	No risk of losing packets during splitting	Physical intervention required + must have enough light
Port mirroring	Easy to configure, most hardware supports it	CPU overhead; Risk of losing packets
Packet broker	Increased flexibility and capability	Dedicated hardware; expense

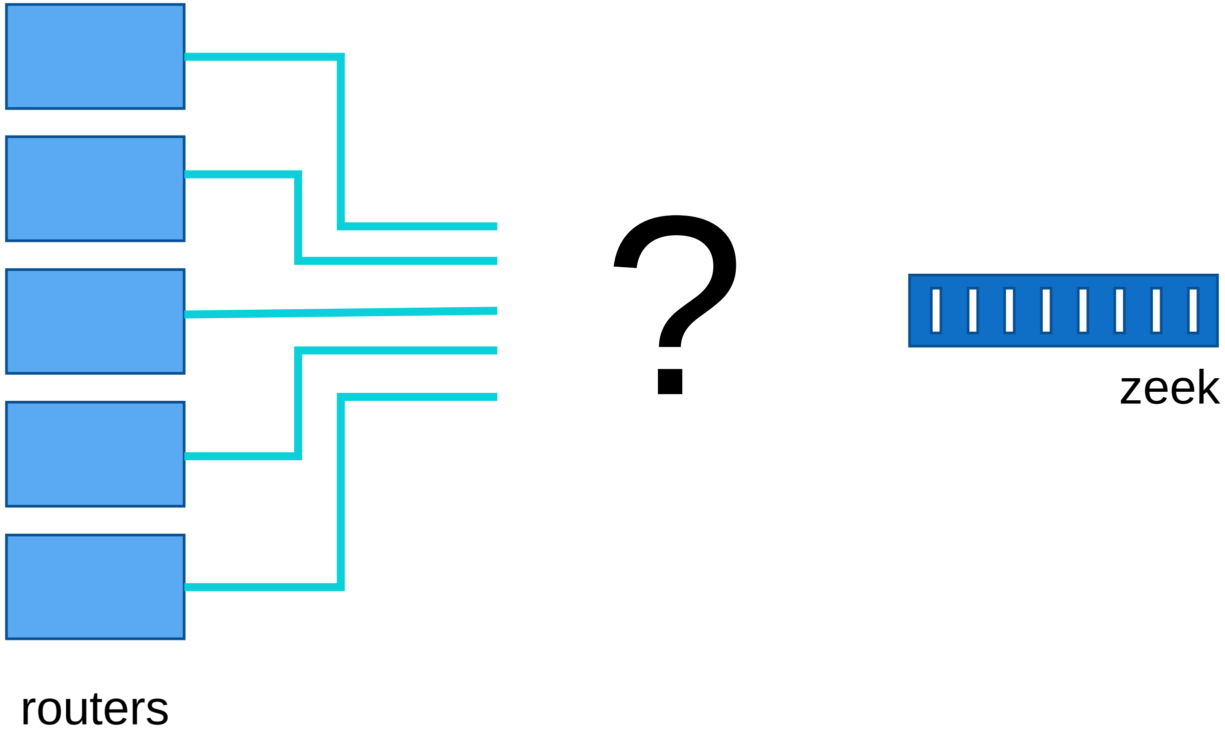
Pros and cons

	Pros	Cons
Optical taps	**No risk of losing packets during splitting**	Physical intervention required + must have enough light
Port mirroring	Easy to configure, most hardware supports it	CPU overhead; Risk of losing packets
Packet broker	Increased flexibility and capability	Dedicated hardware; expense

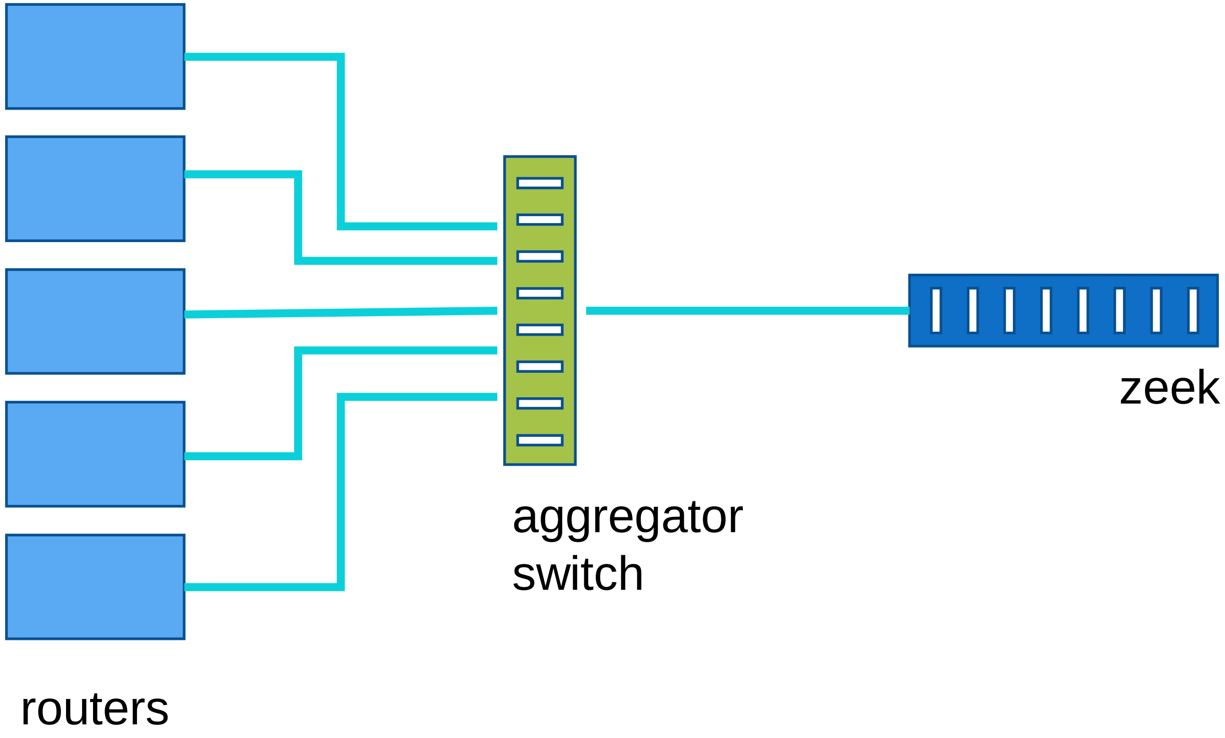
Logical design: taps to Zeek



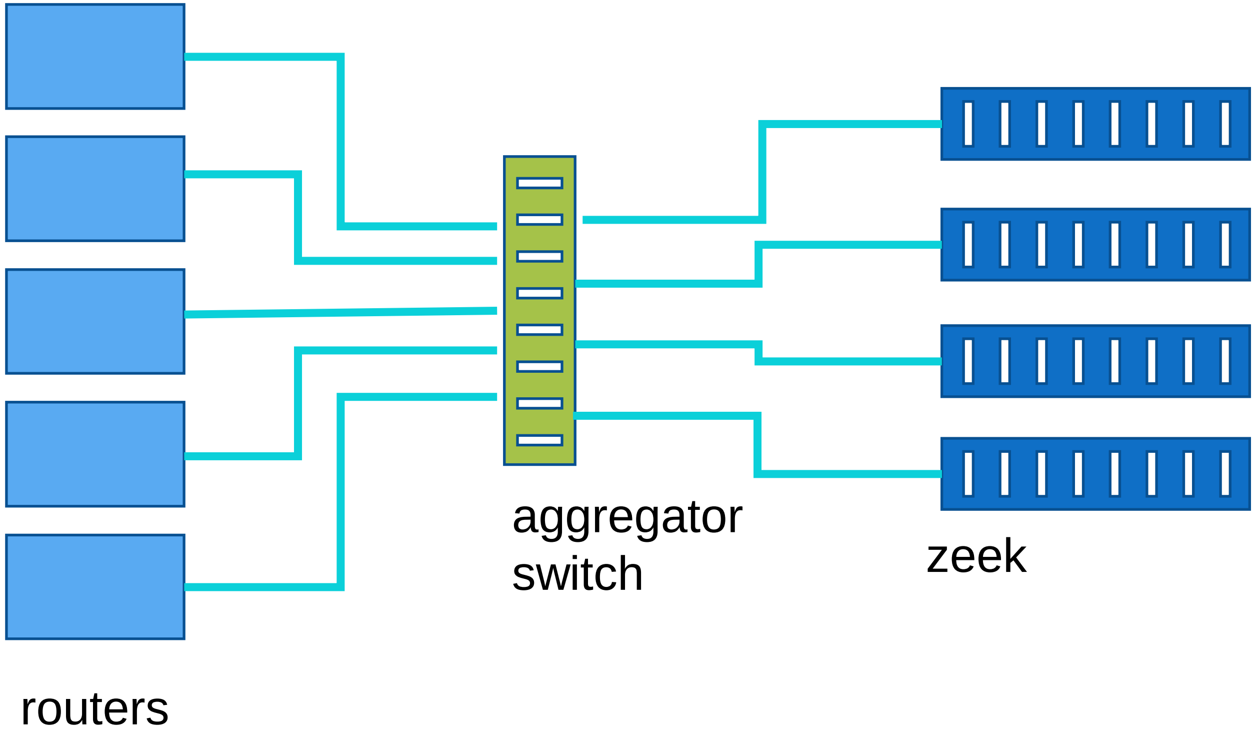
Logical design: taps to Zeek



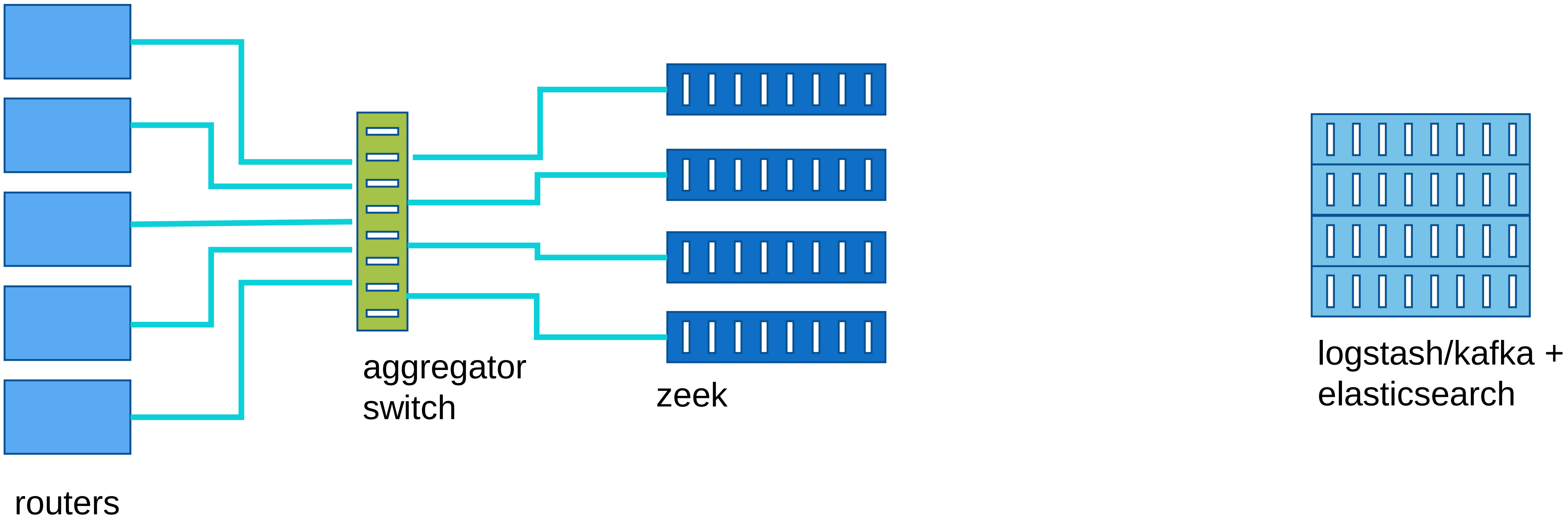
Logical design: taps to Zeek



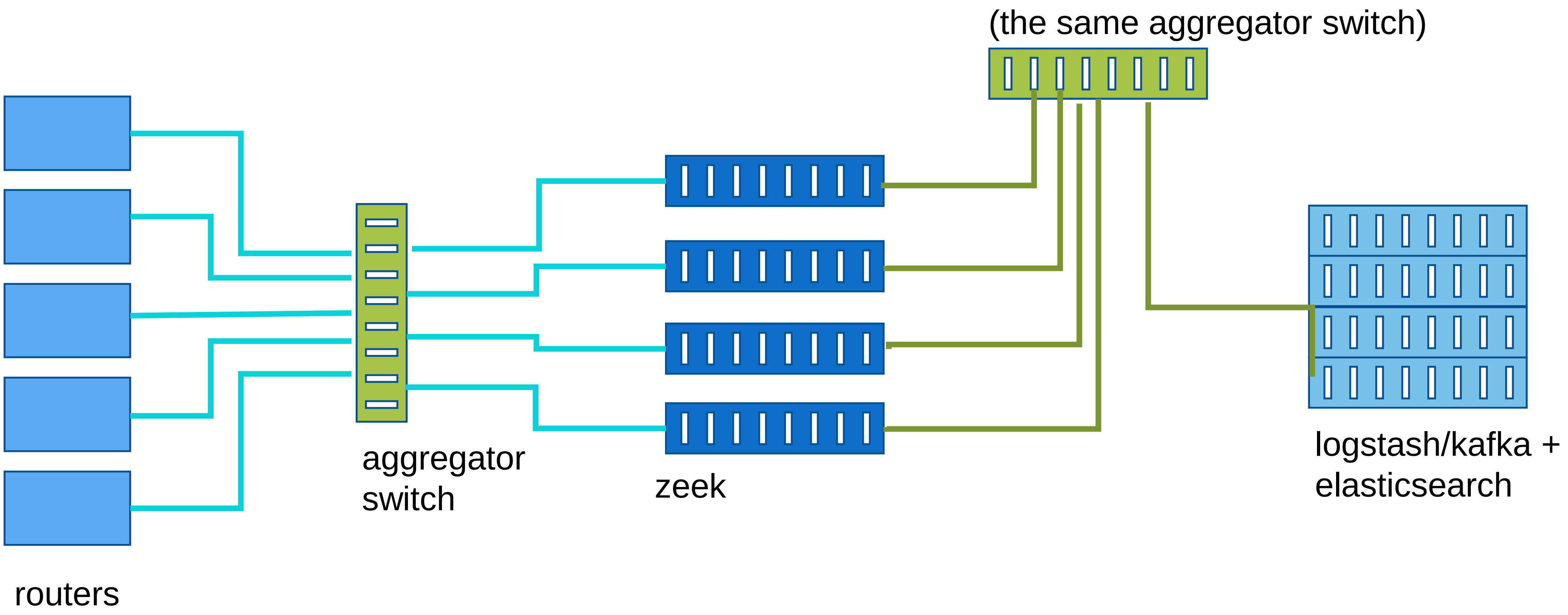
Logical design: taps to Zeek



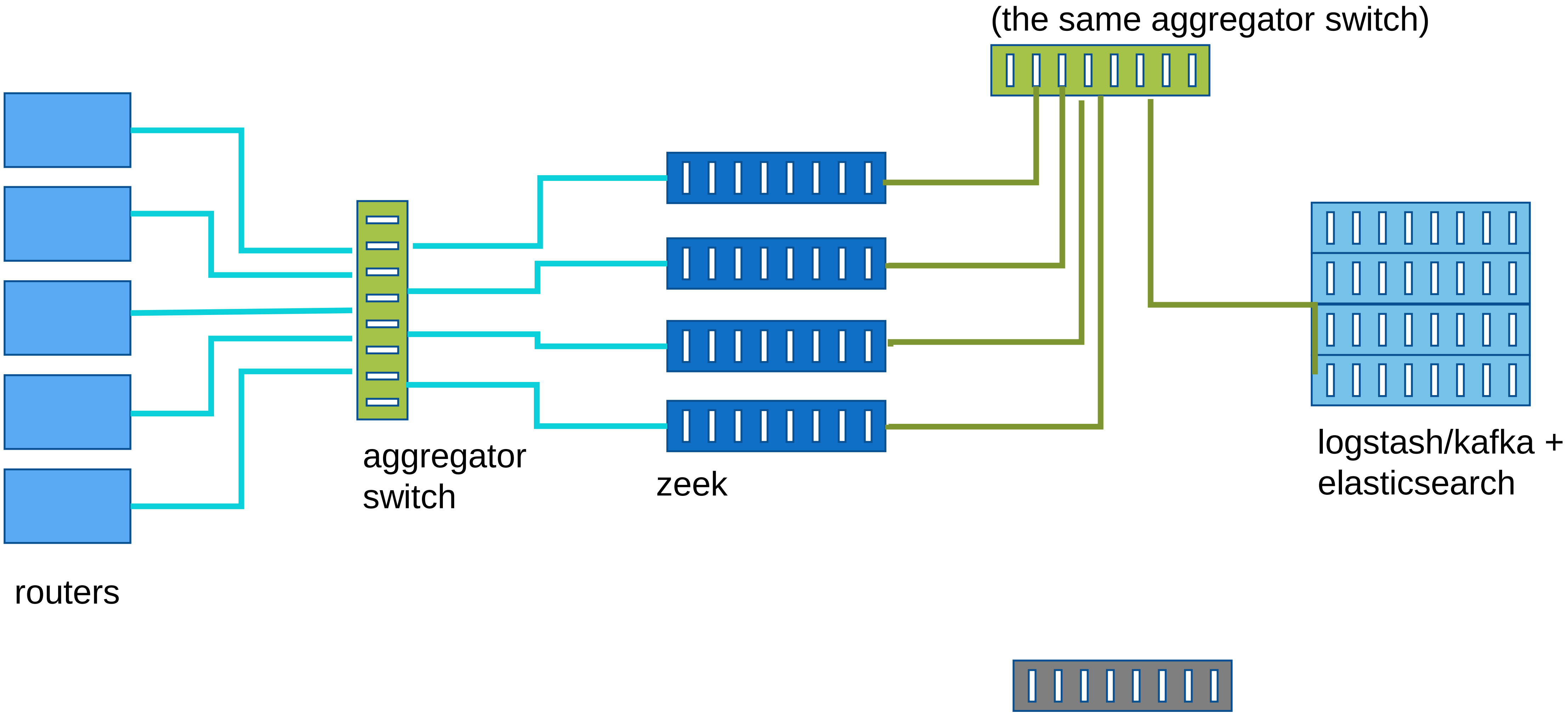
Logical design: Zeek to Elasticsearch



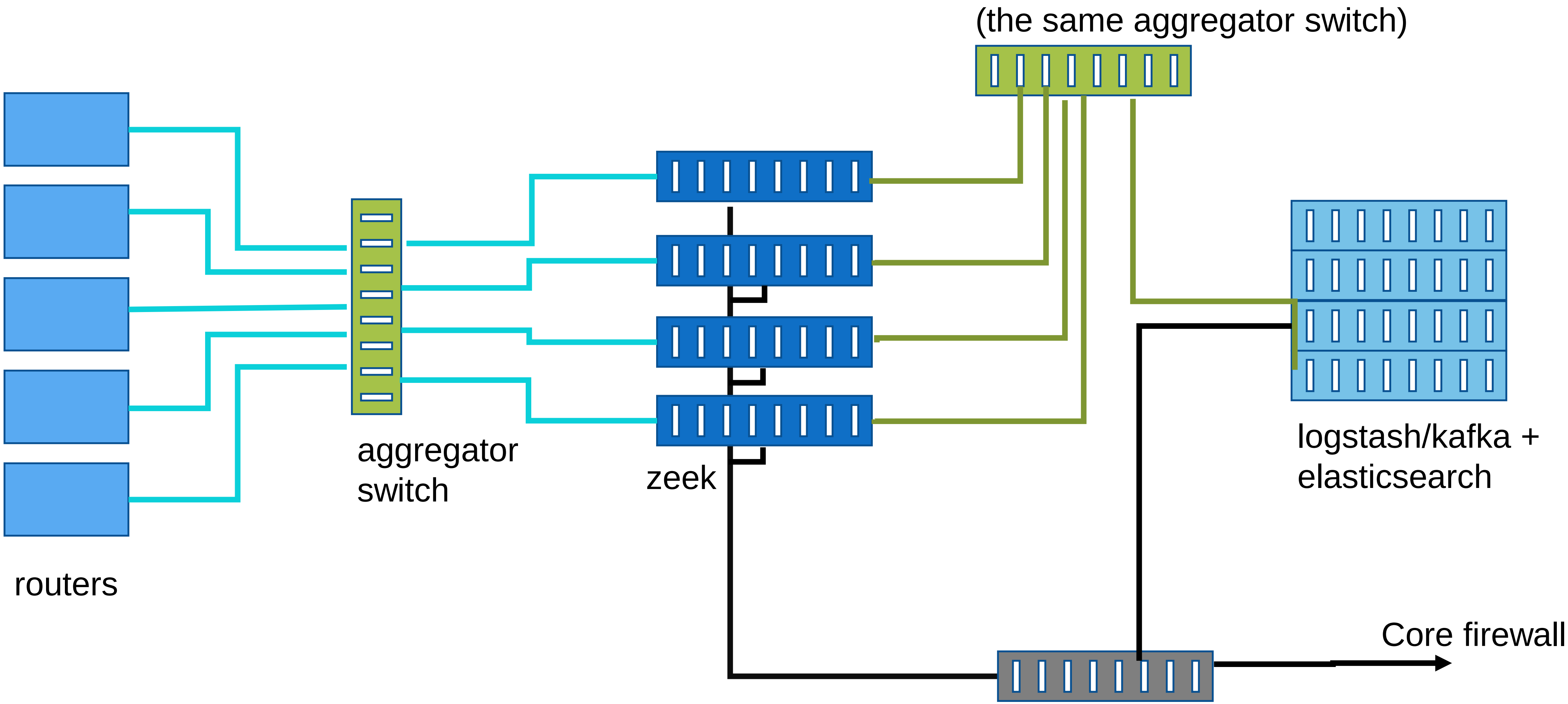
Logical design: Zeek to Elasticsearch



Logical design: admin/operator network

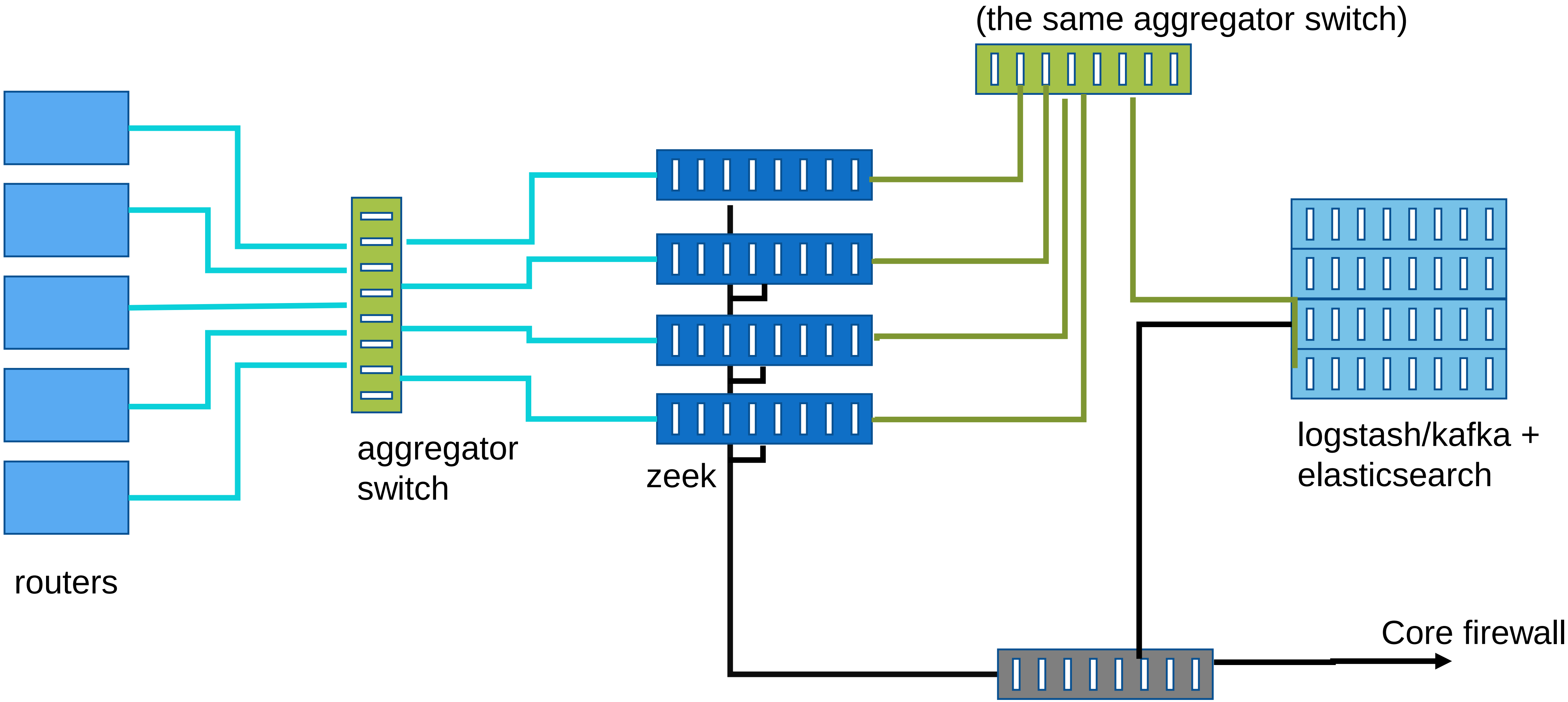


Logical design: admin/operator network



Aggregator management links not shown

Logical design: IPMI network



(the same aggregator switch)

routers

aggregator switch

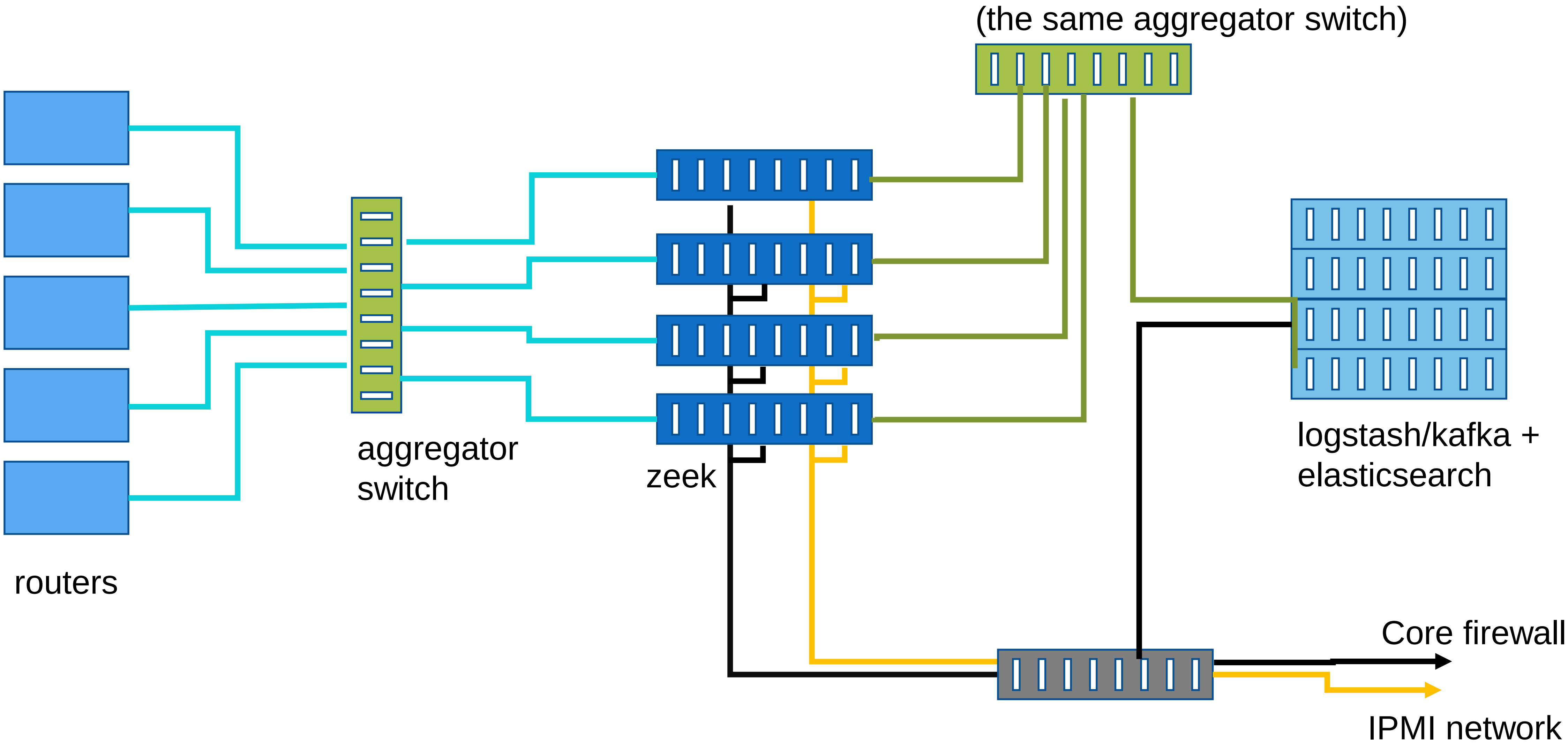
zeek

logstash/kafka + elasticsearch

Core firewall

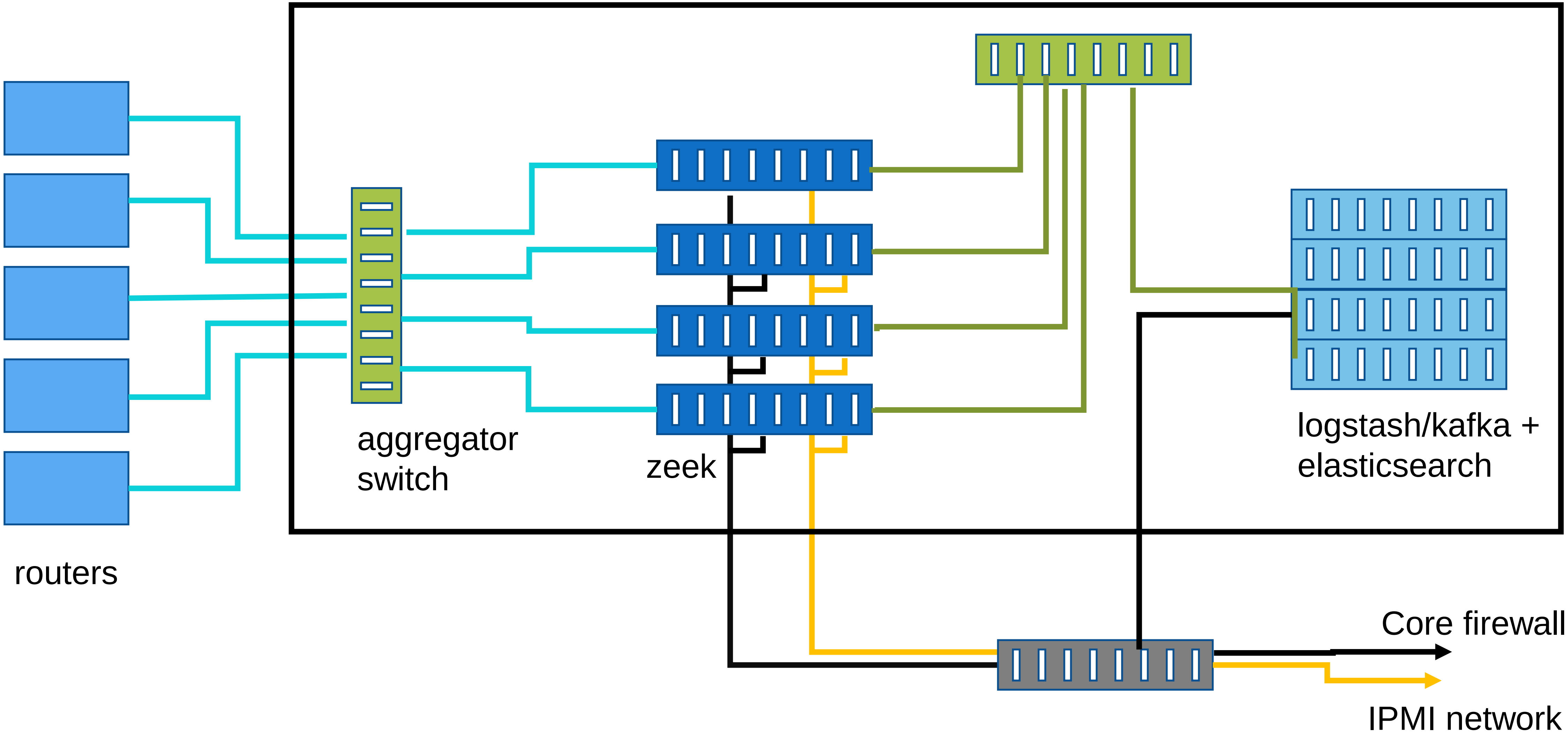
Aggregator management links not shown

Logical design: IPMI network



Aggregator management links not shown

Logical design



Aggregator management links not shown

Physical networks

- Ingest of traffic to aggregator
  - Not really a true network but data capture
- High throughput network from zeek nodes to elasticsearch
- Admin/operator network
  - All traffic routed through core firewall
- Management network

- These last two on two VLANS on one switch

Deployment plan

- The LHCOPN (should!) not contain personal information
  - Beyond that required for authn/z

- Start with this link for initial deployment

Components

- Zeek
  - SOC cluster
- MISP
  - Elsewhere on network
- Logstash/Kafka
  - SOC cluster
- Elasticsearch
  - SOC cluster

Deploying Zeek

- Last time discussed that Zeek scales by spreading load over a zeek farm

- Could deploy one manager node **or** with aggregation switch deploy multiple single node farms
- Each worker node receives split of the data and processes it, with one zeek process per core
  - Zeek is single threaded

Zeek specification

- When designing a zeek worker node, what are the main factors?

- Zeek works by:
  - splitting the traffic across cores
  - running a set of internal protocol analysers against each packet
  - running a set of scripts on top of these

Kernel traffic splitting

- We currently use AF_PACKET to split traffic within the kernel

- Zeek module exists which can use this interface to split traffic across the cores available to it
  - Built into zeek > v6.0

Future possibilities: DPDK

- Data Plane Development Kit

- Some success with this in ESNET with reportedly improved performance
    - “Don’t use in production”

- Something to watch

Zeek specification

- What are the implications of the way Zeek works on its hardware needs?

- High core count to split traffic across

- Enough memory to run the scripts you want

Zeek specification

- I/O not such a high concern as logs are many times smaller than original traffic

- Zeek does **not** store a copy of the original packets by default!
  - In case someone asks you ☺

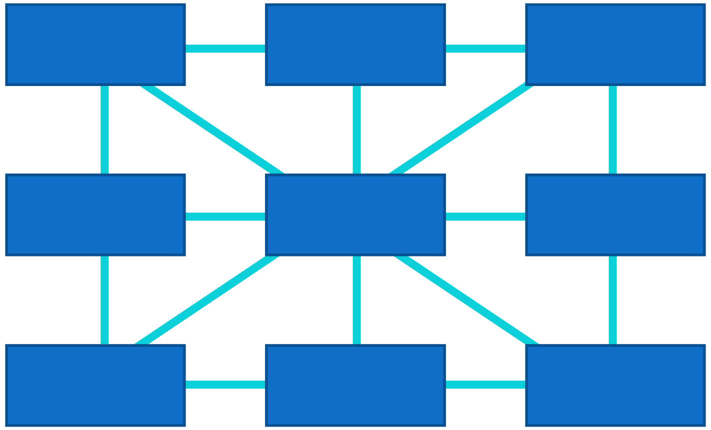
MISP sharing

- Multiple ways of setting up MISP sharing networks
- Here assume multiple synching instances

MISP sharing: Mesh

- Multiple ways of setting up MISP sharing networks
- Here assume multiple synching instances

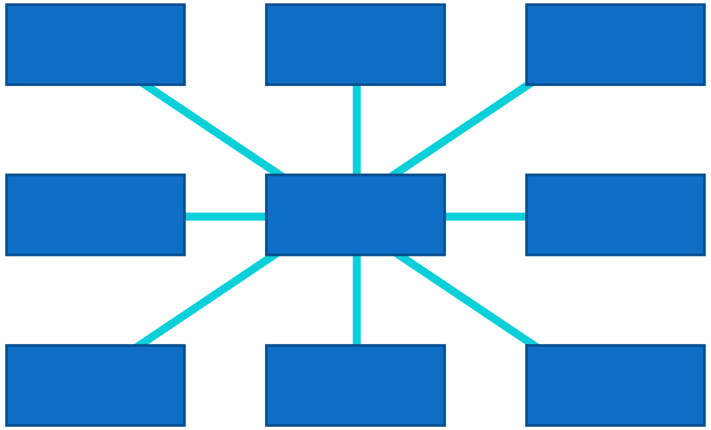
- Mesh topology
  - Every instance is only one hop from ever other instance
  - Becomes very complex very quickly



MISP sharing: Hub and spoke

- Multiple ways of setting up MISP sharing networks
- Here assume multiple synching instances

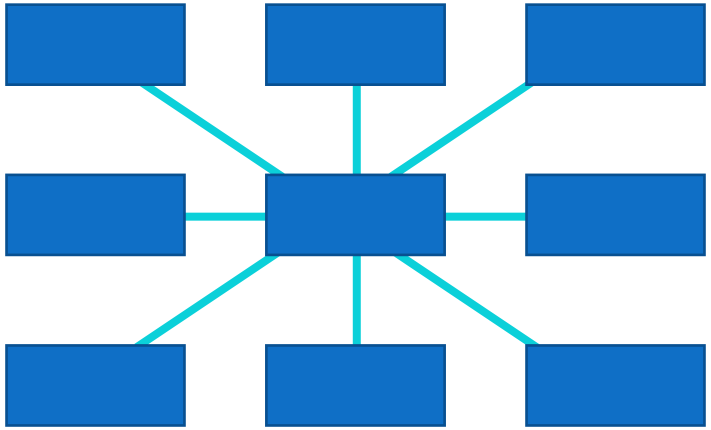
- Hub and spoke topology
  - Relies on central instance for coordination
  - Straightforward to scale
    - Monitoring load on central node



MISP sharing: Hub and spoke

- Multiple ways of setting up MISP sharing networks
- Here assume multiple synching instances

- **Hub and spoke topology**
  - Relies on central instance for coordination
  - Straightforward to scale
    - Monitoring load on central node



How many MISP instances?

- We’ve discussed using MISP to share threat intelligence
- Of course, can also use internally
- CERN has (at least) two instances
  - Internal
  - External sharing instance

How many MISP instances?

- Other organisations use several
  - Internal
  - External
  - Specific instance for malware analysis

- Start simple!

How many MISP instances?

- What does a site need?

- **Primarily** ingesting data:
  - API

- Generating IoCs/sharing with others
  - Web instance

Correlation of traffic and intelligence

- We have Zeek installed, and have our MISP instance/API set up

- How do we do the correlation?

- This time focus on integration at the Zeek/MISP layer

Correlation of traffic and intelligence

- Zeek has an intelligence framework that allows it to perform correlation as it processes data

- We’ll use this to proceed

- In the workshop we’ll see an example of elasticsearch integration

MISP API calls

- Using curl to access MISP API

```
curl --header "$AUTH_KEY" \
     --header "Accept: $JSON" \
     --header "Content-type: $JSON” \
     -X POST \
     --data "{\"request\": \"type\": \"all\"}}" $FEED_URL
```

- Use this to populate a file, `/feed/intel.txt`

Zeek intelligence framework

- The following extract tells Zeek to expire intel after 20 minutes…
  - Why?

- And raise a notice when it matches an indicator of compromise

Zeek intelligence framework

```
@load frameworks/intel/seen
@load frameworks/intel/do_expire

redef Intel::item_expiration = 20min;

const feed_directory = "/feeds";

redef Intel::read_files += {
# MISP feeds
        feed_directory + "/intel.txt",
};

@load
policy/frameworks/intel/do_notice.zeek
```

Alerting

- You can configure Zeek to alert you when it raises particular kinds of notices

- The following is an extract from the main STFC Zeek config that sets this up

Alerting

- STFC uses OpsGenie and – at this stage – our integration works by having Zeek trigger an external script

- Ideally want to fully integrate with Zeek **but** this does show the most general case

Alerting

```
@load ./opsgenie.zeek

hook Notice::policy(n: Notice::Info)
{
    if ( n$note == Intel::Notice )
    {
    add n$actions[Notice::ACTION_OPSGENIE];
    }
```

Alerting

```
function opsgenie_send_notice(message: string)
    {

    when ( local result = 
Exec::run([$cmd=fmt("/usr/local/sbin/opsgenie_alert.sh %s", 
safe_shell_quote(message))]) )
                {
                if ( result$exit_code != 0 )
                        {
                        Reporter::warning(fmt("Opsgenie message did not send 
(%s).", message));
                        return;
                        }
                }

    }
```

SOC roles

- We spoke yesterday about the need for a dedicated team to work on a SOC

- Key roles
  - SOC Service Manager
    - Deployment/Maintenance
  - SOC Analysts
    - Making sense of the data
  - Incident Responders

- These roles can spread across several people!

SOC deployment summary

- We’ve spoken about
  - Physical/logical design
  - Zeek specification
  - MISP sharing topology
  - MISP/Zeek integration
  - Zeek Alerting
  - SOC roles

- Tomorrow in the workshop we will have a chance to look at some of these hands-on

Conclusions: Detection

- In this three lecture block we've looked at
  - The basics of logging, and logging technologies
  - The importance of identifying the most useful logs to avoid "data as noise"
  - The difference between flow based and deep packet inspection network monitoring

Conclusions: Detection

- We’ve also discussed
  - The importance of sharing threat intelligence for our community
  - Tools to help share intelligence responsibly
  - The MISP platform
  - Components of a SOC

Conclusions: Detection

- Finally we've worked through
  - The architecture of an initial SOC deployment
  - The specification for zeek hardware
  - Some detailed zeek configuration
  - SOC roles

- See you tomorrow for the workshop!

Questions?