NETWORK AUTOMATION FORUM

Paradigms of Trust

Architecting Network Automation with Software Engineering Principles - Leandro Lerena, Software Architect - zebbra AG

AUTOCON 5

THE NETWORK AUTOMATION CONFERENCE

The issue with scripts

- Parameter validation
- Connection handling
- Credentials
- Vendor support
- Error handling
- Logging
- Versioning
- Persist Result
- Testing

```python
HOST, USER, PWD  = env["HOST"], env["NET_USER"], env["NET_PWD"]
creds = {"host": HOST, "username": USER, "password": PWD}

try:
   if VENDOR == "cisco_ios":
       conn = ConnectHandler( device_type="cisco_ios", **creds)
       cmd  = "shutdown" if STATE == "down" else "no shutdown"
       conn.send_config_set([ f"interface {INTERFACE}", cmd])
   elif VENDOR == "nokia_srl":
       conn = ConnectHandler( device_type="nokia_srl", **creds)
       verb = "disable" if STATE == "down" else "enable"
       conn.send_config_set([ "set /interface {INTERFACE} ..."])
   else:
       raise ValueError(f"unsupported vendor: {VENDOR}")

   after = conn.send_command( f"show interface {INTERFACE}")
   conn.disconnect()
   result = {"host": HOST, "iface": INTERFACE, "state": STATE}
   log.info( "changed %s/%s -> %s", HOST, INTERFACE, STATE)

except Exception as e:
   # Exception handling

with open(OUT, "a") as f:
   f.write(json.dumps( result) + "\n")

sys.exit( 0 if result["ok"] else 1)
```

- Unknown safety profile
- Interpretability
- Auditability
- Parallel Execution is risky. In general, race conditions cannot be detected

The issue with scripts

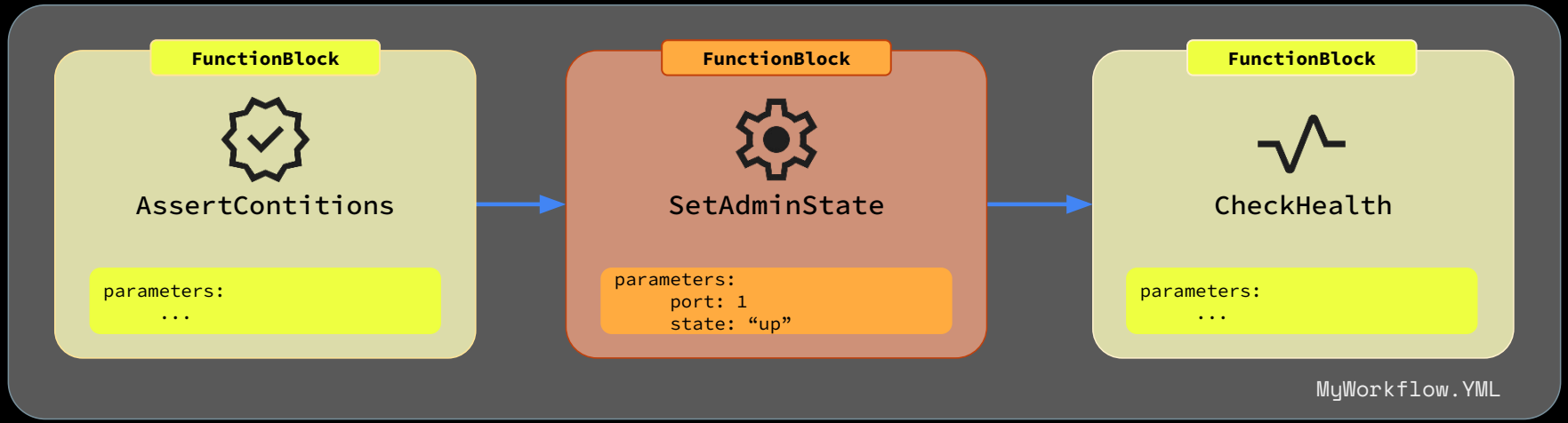
- Parameter validation
- Connection handling
- Credentials
- Vendor support
- Error handling
- Logging
- Versioning
- Persist Result
- Testing
- Implementation & Framework Trust

- Unknown safety profile
- Interpretability
- Auditability
- Parallel Execution is risky. In general, race conditions cannot be detected

- Transactional Trust


NAF
NETWORK AUTOMATION FORUM
Transactional Trust

Transactional Trust: Motivation



Interpretability

What does a failing operation mean? Is it reason to escalate or can we analyze it calmly tomorrow?

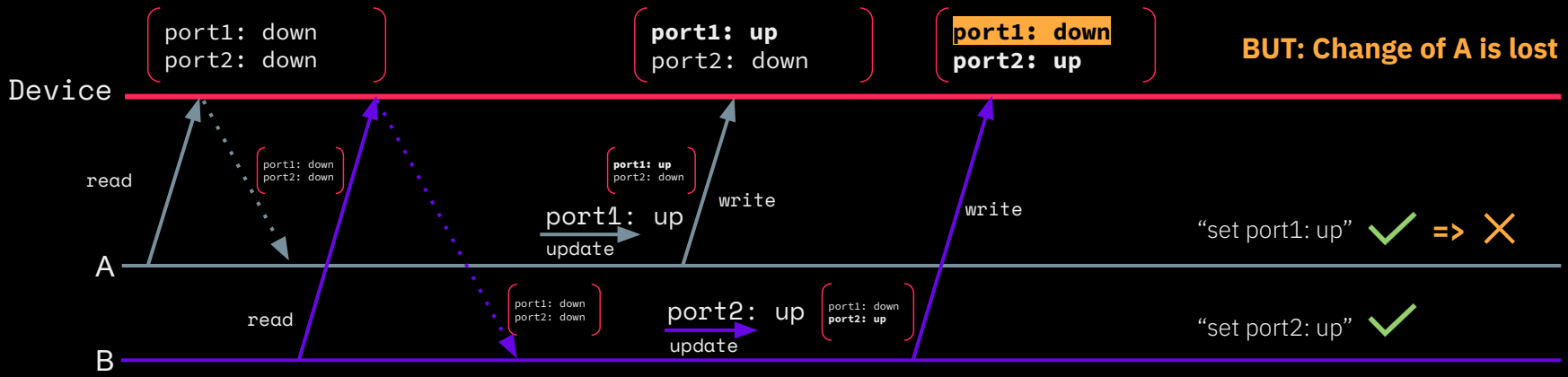
Auditability

What operations did (potentially) touch a device? What was the intent?

Race conditions

Even when executing simple scripts, race conditions may occur, leading to **inconsistent state.**

Transactional Trust: Race condition



Atomicity

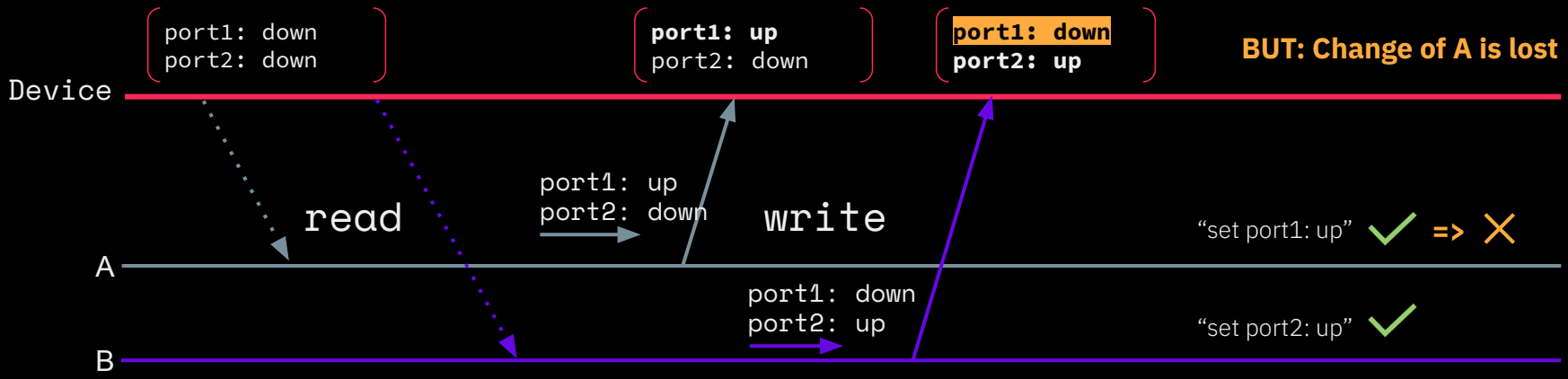
Most operations are not atomic by nature. Complex operations can often not be formulated atomically

Need for central orchestrator

A central orchestrator can mitigate these issues.

Problems are already solved in Relational Databases!

Transactional Trust: Race condition



Atomicity

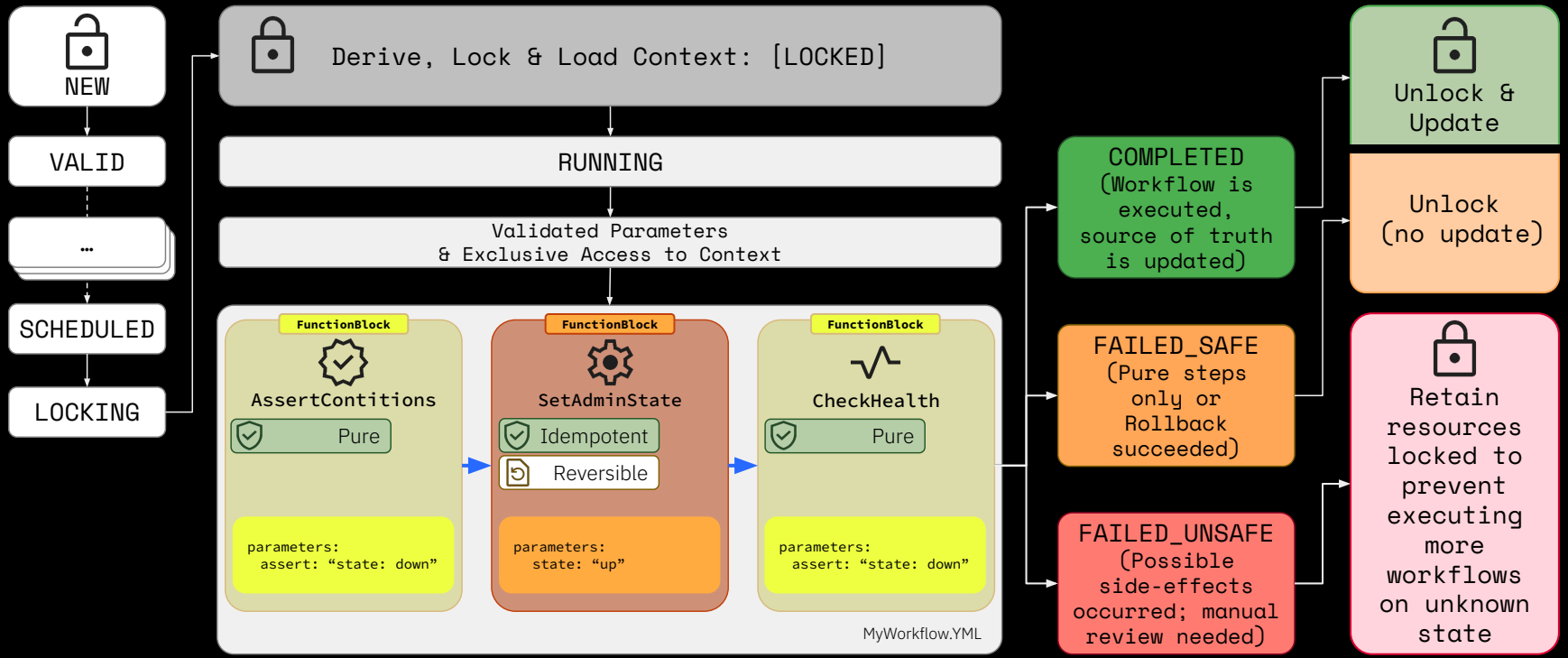
Most operations are not atomic by nature. Complex operations can often not be formulated atomically

Need for central orchestrator

A central orchestrator can mitigate these issues.

Problems are already solved in Relational Databases!

Workflow as a Transaction



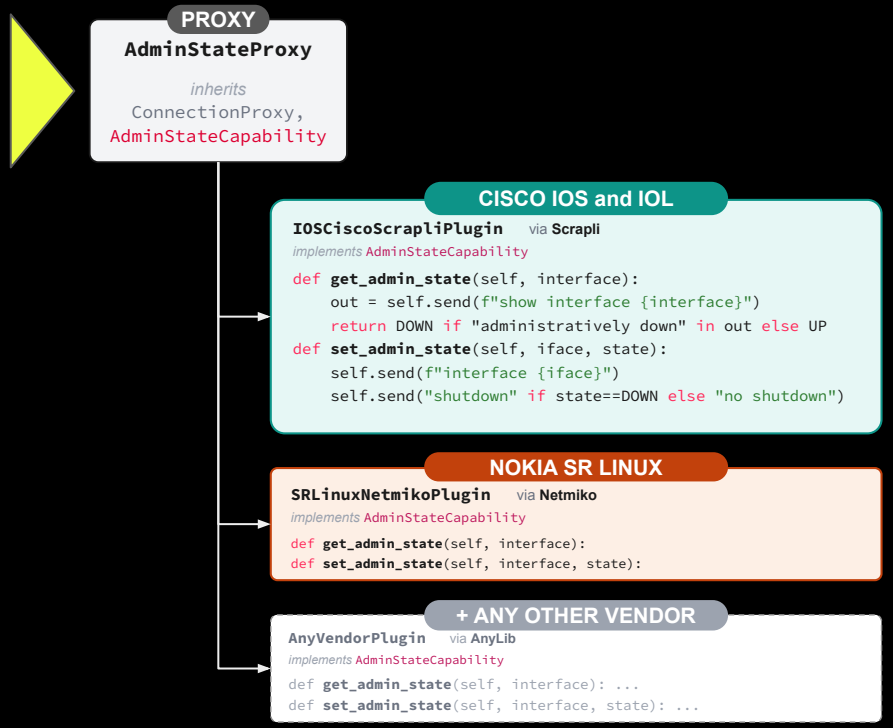

NAF
NETWORK AUTOMATION FORUM
Implementation &
Framework Trust

Framework Trust

```
@fb_test_case_with_lab(...)
@fb_test_case_with_lab(...)
@register_function_block(...)

class SetAdminState(FunctionBlock):
 async def run(self, p, ctx):
   with AdminStateProxy.connect(ctx.device) as conn:
     conn.set_admin_state(p.interface, p.desired_state)
     new_state  = conn.get_interface_admin_state(p.interface)
     return FunctionBlockResult(
     success = after is p.desired_state,
     data    = SetAdminStateResult(p.interface, new_state)
     )
```

Separate business logic from vendor-specific implementation



Framework Trust

```
@fb_test_case_with_lab(...)
@fb_test_case_with_lab(...)

@register_function_block(
   name           = "set_admin_state",
   version        = (1, 0, 0),
   run_on         = "device",
   is_idempotent = True, # if it fails, the orchestrator can
retry!
)

class SetAdminState(FunctionBlock):
   ...
```

Function properties:

- Semantic versioning
- Scope
- Safety profile
  - *Pureness*
  - *Idempotency*
  - *Reversible*

Implementation Trust

```
@fb_test_case_with_lab(”Shut IF on Cisco IOL",
   remote_lab_fixture=simple_iol,
   params=SetAdminStateParams(interface="Ethernet0/1",
                              desired_state=AdminState.DOWN),
   assertions=[lambda r: r.data.new_state == dminState.DOWN],
)

@fb_test_case_with_lab(”Bring IF up on Nokia SR Linux",
   remote_lab_fixture=simple_srlinux,
   params=SetAdminStateParams(interface="ethernet-1/1",
                              desired_state=AdminState.UP),
   assertions=[lambda r: r.data.new_state == AdminState.UP],
)

@register_function_block(...)
class SetAdminState(FunctionBlock):
   ...
```

CISCO IOL

```
=== Testing: Shut Ethernet0/1 on Cisco IOL ===
```

NOKIA SR LINUX

```
=== Testing: Shut ethernet-1/1 on Nokia SR Linux ===
```

Powered by Netlab (https://netlab.tools/)

Future-Proofing – Safe Agentic AI

Paradigms for safe Agentic AI

Tier 1 - Autonomous (no risk)

Investigate freely

only function blocks without side-effects (“pure”)

“Read state, gather context, propose a hypothesis. The agent cannot change anything by design.”

Tier 2 - Autonomous (pre-approved)

Act within an allow-list

pre-approved function blocks (pref. idempotent, reversible)

“A narrow set of operations the team marks auto-runnable. Locked context. Transactional rollback if anything goes sideways”

Tier 3 - Supervised

Propose, Engineer accepts

Any avilable function block

“The agent drafts the workflow. A human reviews the declared properties and the locked context, then presses run.”

Safety benefits

No raw operations

Every operation remains structured and auditable

“Agents call function blocks, cannot bypass auditability, logging. Agents are never in contact with device credentials. Clear safety boundaries”

Take-Aways

1 Trust is layered. Name the layers, so you can build them.

Transactional · Implementation · Framework · Future. Each one is something a script cannot give you on its own.

2 Evaluate paradigms, not products.

Transactional · Implementation · Framework · Future. Each one is something a script cannot give you on its own.

3 Well-defined paradigms build the foundation for agents.

Purity, idempotency, locked context, strict schemas. The work compounds. The same boundary protects both.

NETWORK AUTOMATION FORUM

Remote Lab - Developer Preview:
https://github.com/zebbra/remote-lab

- Integrate Netlab directly into your tests via easy-to-use pytest fixtures
- Maintain one lab for multiple engineers and the CI/CD pipeline.

"That last percent for fully automated realistic e2e testing"

engineered
for neops.io
at zebbra.ch

Questions?