Consortium GARR

From Clicks to Code for GARR's Optical Network

Matteo Colantonio, Filippo Landini

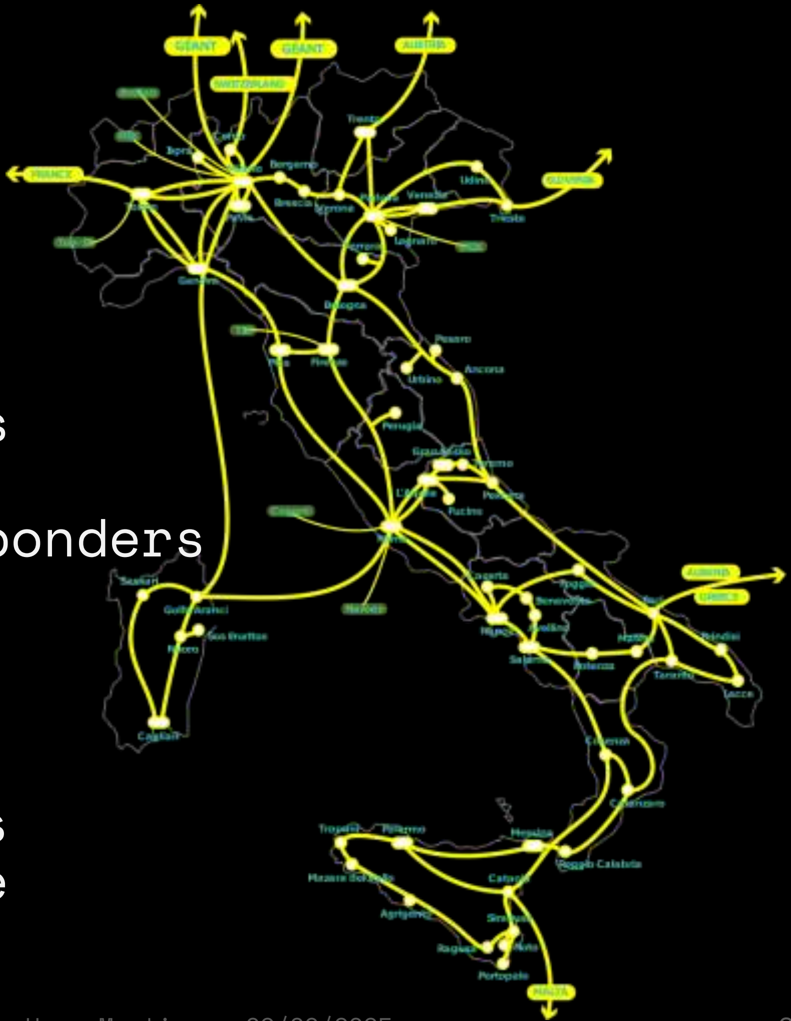
GARR Optical Network

98
add/drop
sites

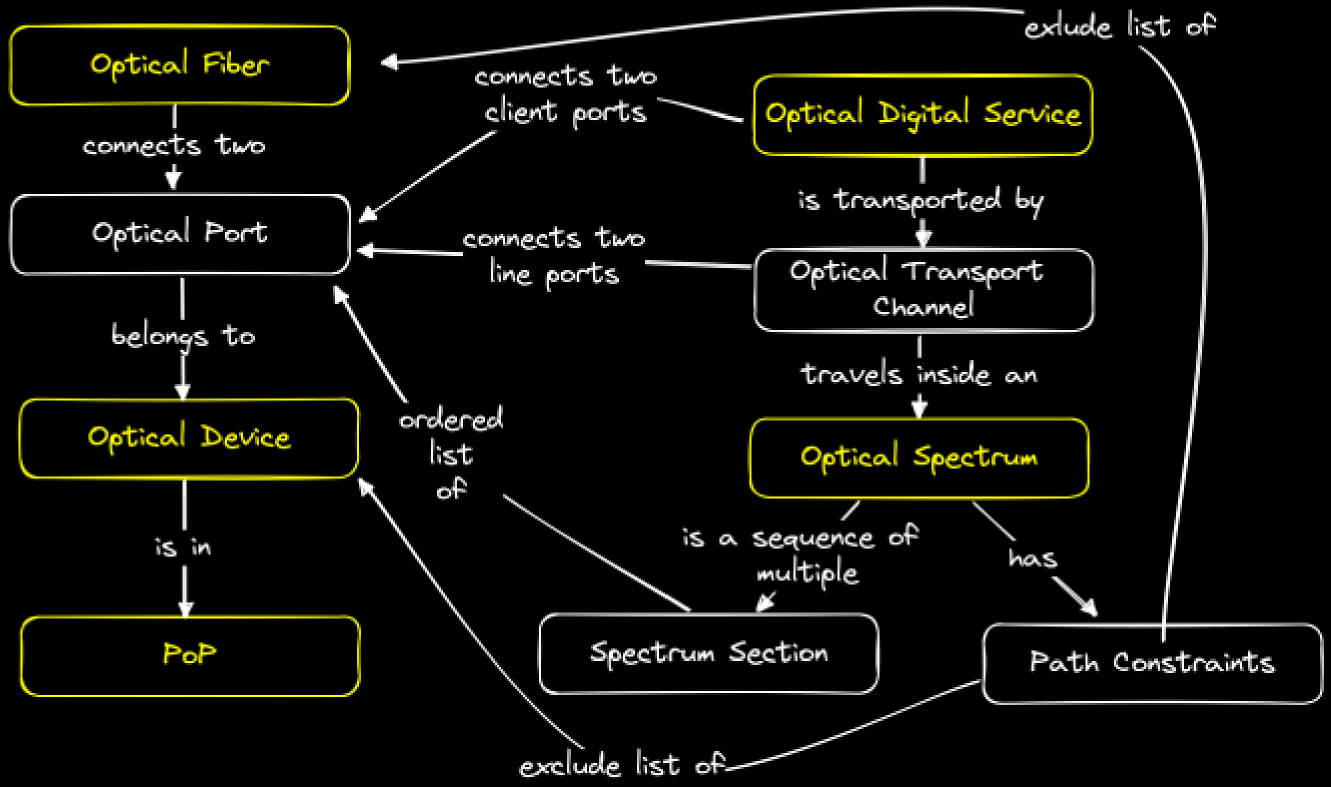
71	ROADMs
84	ILAs
115	Transponders

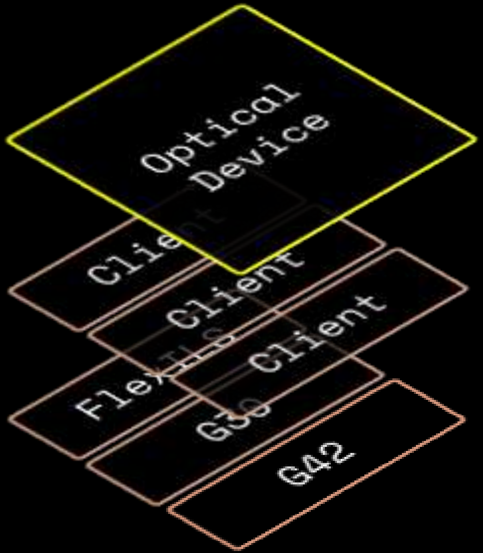
20000 km
of optical
fibers

57.8 Tbps
line-side
capacity



Defining our reality





One function to rule them all

Problem: same task, different platform
For example, in a workflow step:

```
for port in subscription.optical_fiber.terminations:
    device = port.device
    port_name = port.port_name
    set_port_admin_state(device, port_name, "up")
```

Idea: one function with multiple implementations for each platform

```python
@set_port_admin_state.register(Platform.FlexILS)
def _(
    optical_device: OpticalDeviceBlock,
    port_name: str,
    admin_state=Literal["up", "down", "maintenance"],
) -> Dict[str, Any]:
    # FlexILS implementation

@set_port_admin_state.register(Platform.G30)
def _(…same args…) -> Dict[str, Any]:
    # G30 implementation
```

Benefits: keeps code organized, easy to add new platforms, simplifies workflow logic

Keep it simple, Talk Direct

Problem:
- NBI = loss of control/functionalities
- Ansible = just adds complexity

Idea: communicate directly with devices like any other API

Benefits: simplicity and maintainability

```python
@set_port_admin_state.register(Platform.Groove_G30)
def _(
    optical_device: OpticalDeviceBlock,
    port_name: str,
    admin_state=Literal["up", "down", "maintenance"],
) -> Dict[str, Any]:
    ids = port_name.split("-")[-1]  # port-1/2/3 -> 1/2/3
    shelf_id, slot_id, port_id = ids.split("/")  # 1/2/3 -> 1, 2, 3

    g30 = g30_client(optical_device.mngmt_ip) # RESTCONF client
    port = g30.data.ne.shelf(shelf_id).slot(slot_id).card.port(port_id)
    # dynamic Path → https://{{host}}:{{port}}{{+restconf}}/data/ne:ne/shelf={{
shelf_id}}/slot={{slot_id}}/card/port={{port_id}}

    port.modify(admin_status=admin_state) # PATCH method with data validation

    return port.retrieve(depth=2) # GET method
```

```python
@set_port_admin_state.register(Platform.GX_G42)
def _(
    optical_device: OpticalDeviceBlock,
    port_name: str,
    admin_state=Literal["up", "down", "maintenance"],
) -> Dict[str, Any]:
    shelf_id, slot_id, port_id = port_name.split("-")  # 1-4-L1 -> 1, 4, L1

    g42 = g42_client(optical_device.mngmt_ip)
    port = g42.data.ne.equipment.card(f"{shelf_id}-{slot_id}").port(port_id)

    port.modify(admin_state=admin_state)

    return port.retrieve(depth=2)
```

Demo

We just started…

- validate, terminate wf
- K8s deployment
- long running steps live feedback

Thanks