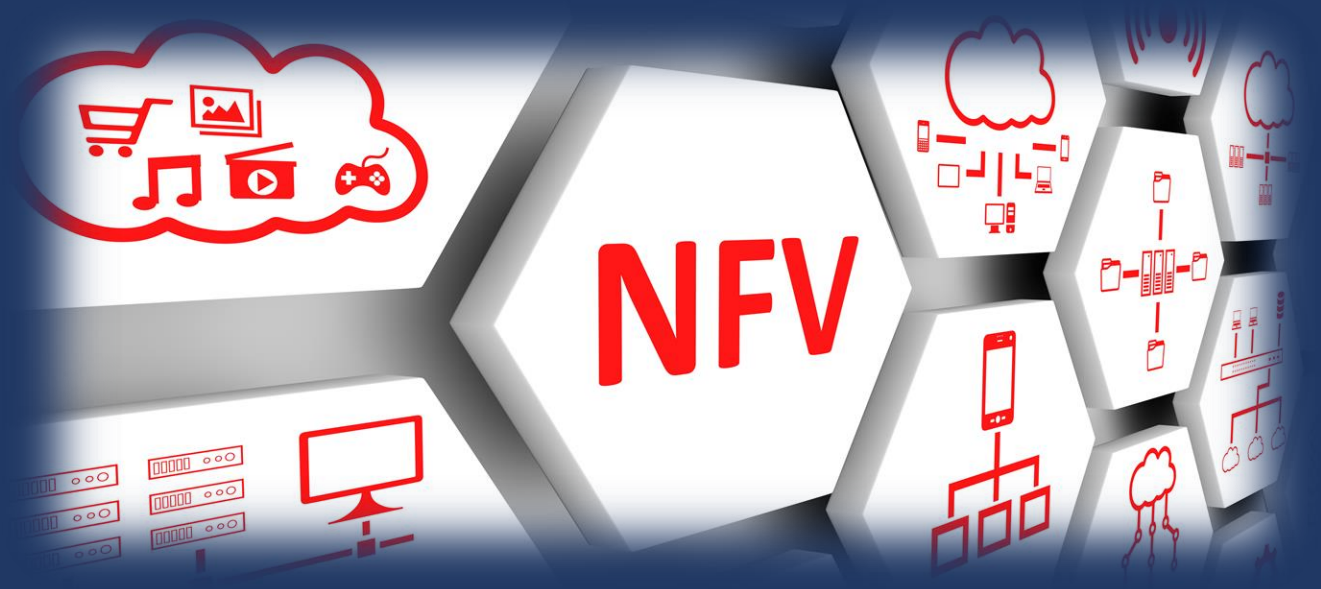
DIPARTIMENTO DI ELETTRONICA
INFORMAZIONE E BIOINGEGNERIA
POLITECNICO
MILANO 1863

Project-Team 5
Amirhosein Ataei
Seyedeh Mona Sabeti Razavian



Smart Networks and Service Orchestration

OLS T-API Topology Service

OLS **Topology Service parsing** via ONF T-API over RESTCONF

Politecnico di Milano, May 30th 2020

ADVA™

iWAY Engineering Services

H2020 Metro Haul

PEoPLe @DEIB
Politecnico Experiences on Passionate Learning

Outline

- Objectives:
  - Parse the ONF T-API topology
  - Get T-API context, inventory, and topology
  - **Iteratively parse nodes and links and build up routing topology information**
  - Performs **DIJKSTRA** shortest path routing

- Steps:

  1 . Get T-API context

  2 . Get list of nodes

  2.1 . Iteratively query individual nodes

Understanding

- **Now we consider some concepts that is so important for continue**

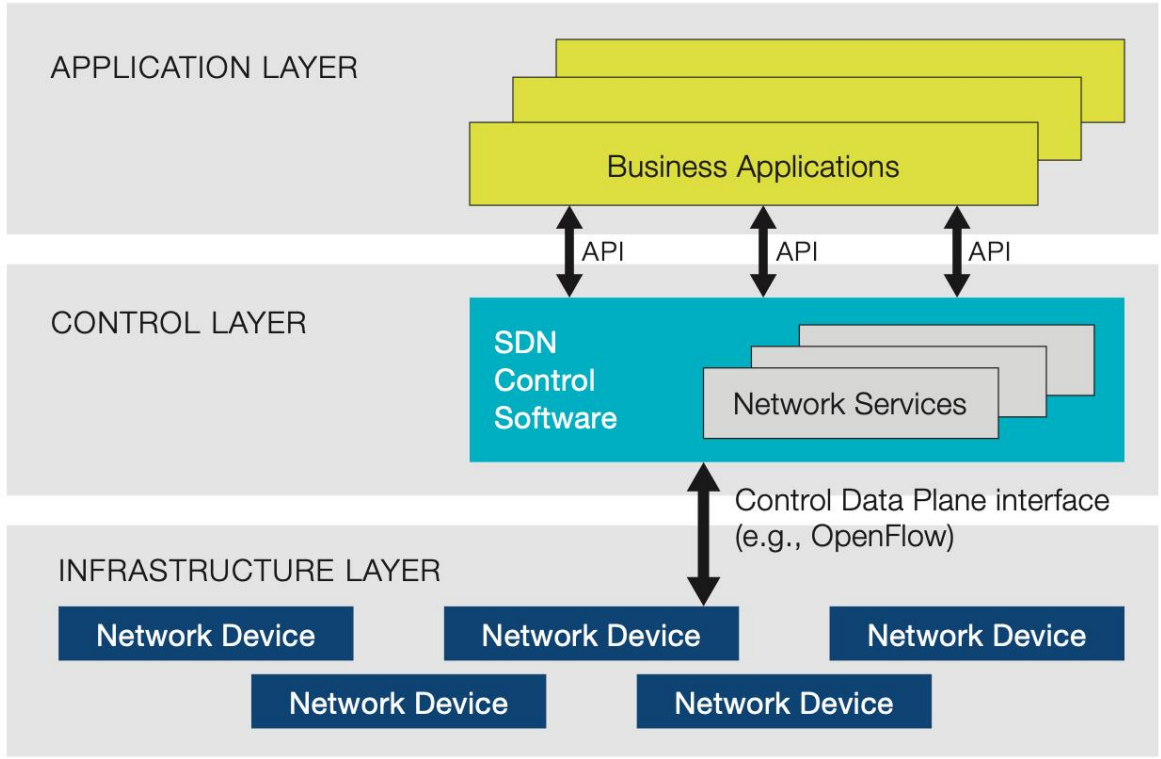
SDN ?

architectural components:

- **SDN Application**
- **SDN Controller**
- **SDN Datapath**

  SDN Control to Data-Plane Interface (CDPI)

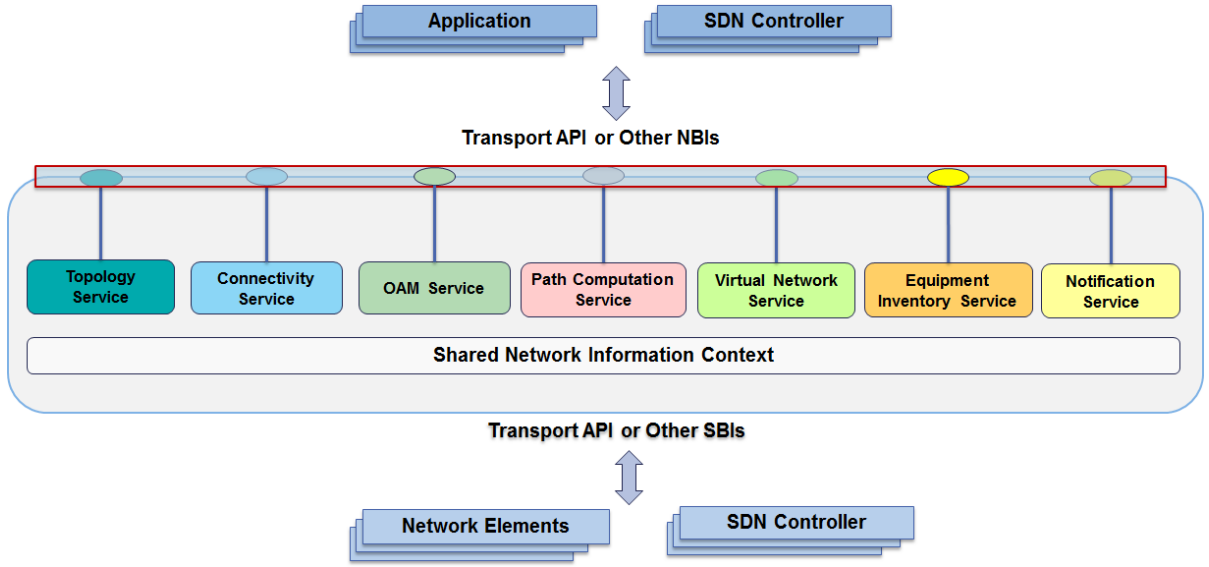
  SDN Northbound Interfaces (NBI)



- https://www.youtube.com/channel/UCenU1k_-u08B25eXWbuvVrA

T-API ?

- T-API (Transport API) is a standard API developed by the Open Networking Foundation (ONF) that allows a T-API client (e.g. carrier's orchestration platform or a customer's application) to retrieve information from and control a domain of transport network equipment controlled by a T-API server (e.g. Transport SDN Controller).

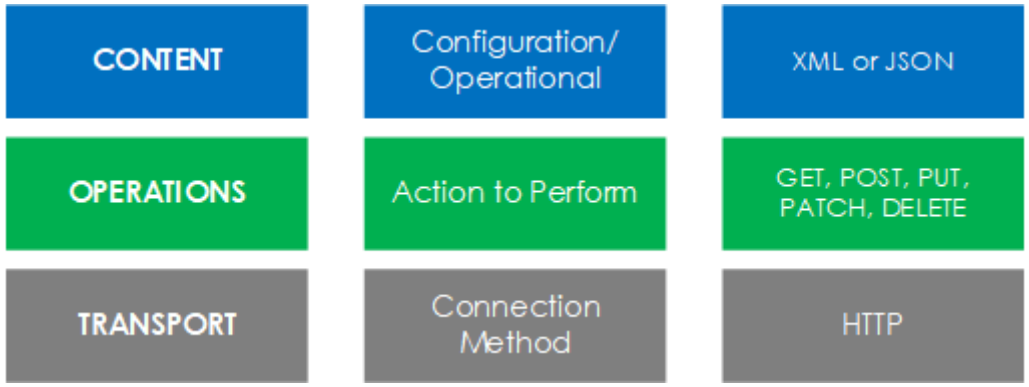


- https://wiki.opennetworking.org/display/OTCC/TAPI

RESTCONF ?

First let's look at the protocol stack and step through the layers,

- Content - Unlike NETCONF where we much use XML. RESTCONF allows for either JSON or XML to be used.
- Operations - Each of the operations are aligned to the various HTTP methods, providing the required suite of CRUD based operations (Create, Replace, Update and Delete).
- Transport - The transport protocol is HTTP, allowing us to use HTTPS. Providing the security benefits that TLS has to offer.



JSON ?

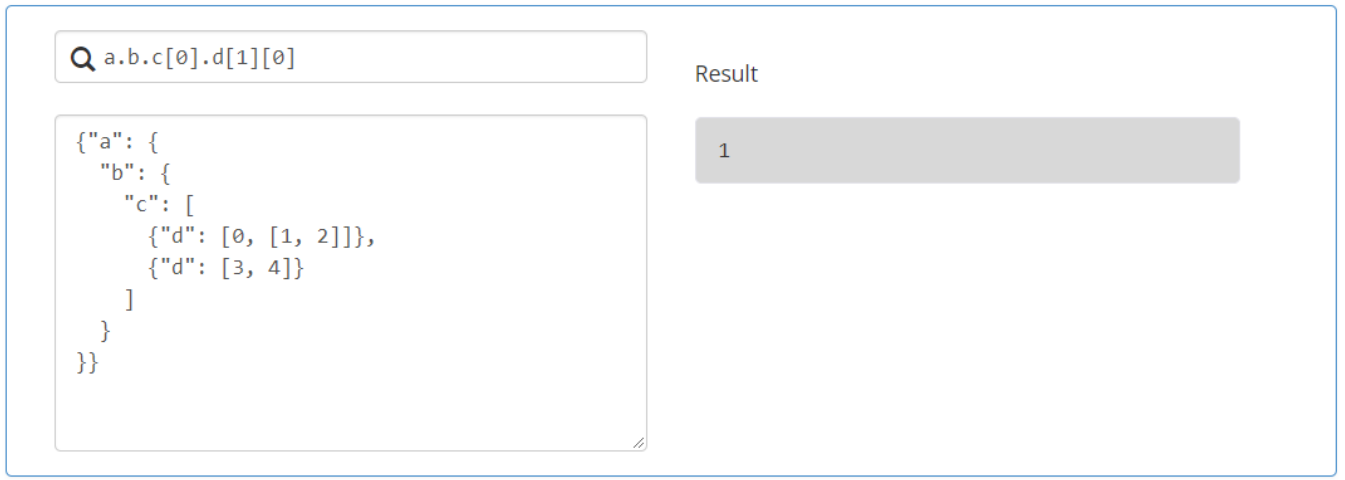
- **JSON** (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

- **JSON is built on two structures:**
  - A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
  - An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

JMESPath ?

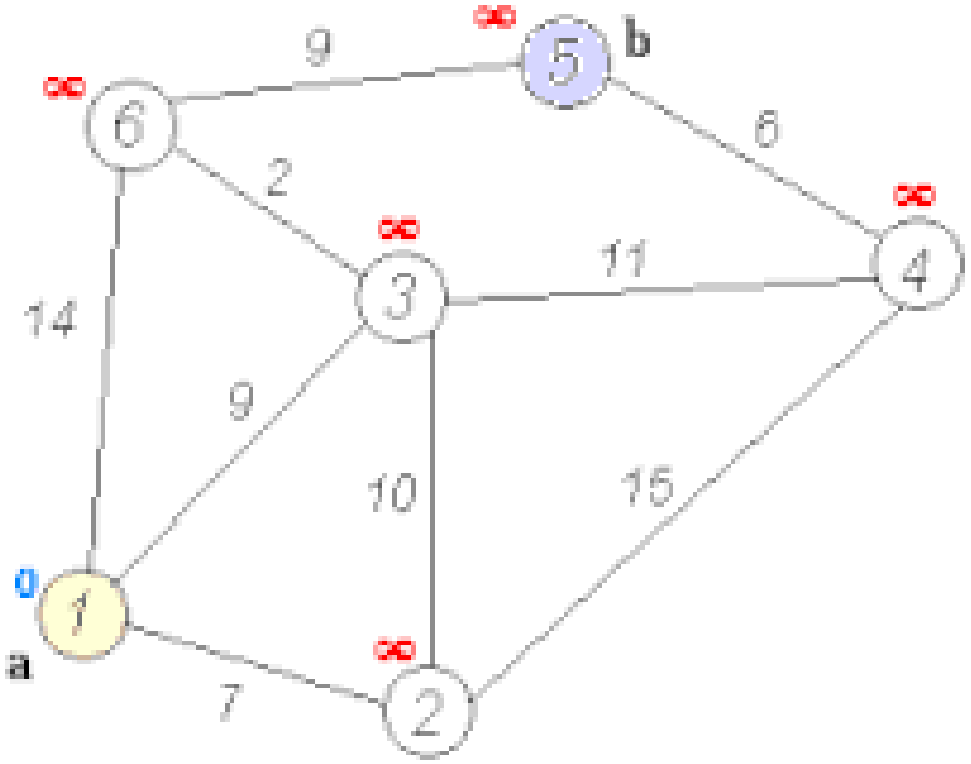
- JMESPath is a query language for JSON. You can extract and transform elements from a JSON document.



- https://jmespath.org/tutorial.html

DIJKSTRA ?

- Dijkstra's algorithm to find the shortest path between *a* and *b*. It picks the unvisited vertex with the lowest distance, calculates the distance through it to each unvisited neighbor, and updates the neighbor's distance if smaller. Mark visited (set to red) when done with neighbors.



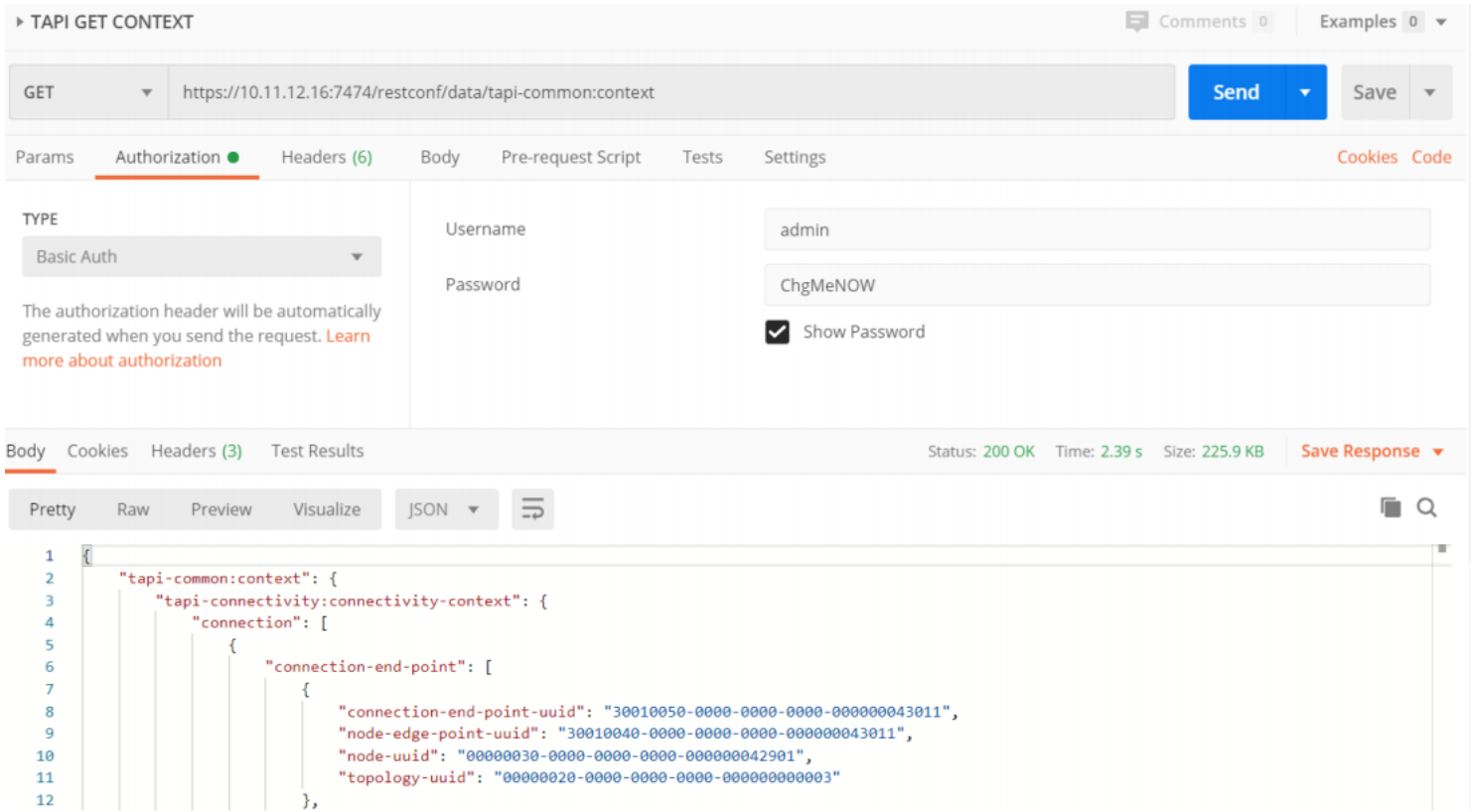
Real Test

- **Now we go to code section to implement our concept into reality**

(connect to polimi vpn and go on)

Get T-API context

- Step 1:
- GET https://10.11.12.16:7474/restconf/data/tapi-common:context



GET Node UUID list

- Step 2:
- GET http://10.11.12.16:7474/restconf/data/tapi-common:context/tapi-common:service-interfacepoint?fields=uuid


```
{
    "tapi-topology:node": [
        {
            "uuid": "00000030-0000-0000-0000-
000000042901"
        },
        {
            "uuid": "00000030-0000-0000-0000-
000000043211"
        },
        {
            "uuid": "00000030-0000-0000-0000-
000000042665"
        }
    ]
}
```

Iteratively query individual nodes

- Step 2.1:
- GET https://10.11.12.16:7474/restconf/data/tapi-common:context/tapi-topology:topology-context/topology= 00000020-0000-0000-0000-000000000003/node=00000030-0000-0000-0000-000000042901


```
{
    "tapi-topology:node": [
        {
            "administrative-state": "UNLOCKED",
            "tapi-adva:adva-node-spec": {
                "ne-type": "FSP 3000R7",
                "reachability-status": "REACHABLE",
                "serial-number": "FA71192952135",
                "sw-version": "19.1.2",
                "user-label": "Nodo 1 Rack basso"
            },
 "layer-protocol-name": [
                "PHOTONIC_MEDIA",
                "DSR",
                "ODU"
            ],
 "lifecycle-state": "INSTALLED",
            "name": [
                {
                    "value": "/Nodo 1 Rack basso",
                    "value-name": "USER"
                },
…            ],
            "operational-state": "ENABLED",
                        "owned-node-edge-point": [
                            {
                                "administrative-state": "UNLOCKED",
                                "tapi-connectivity:cep-list": {},
                                "layer-protocol-name": "PHOTONIC_MEDIA",
                                "lifecycle-state": "INSTALLED",
                                "link-port-direction": "BIDIRECTIONAL",
                                "mapped-service-interface-point": [
                                    {
                                        "service-interface-point-uuid": "00000010-0000-0000-0000-000000043257"
                                    }
                                ],
                                "tapi-photonic-media:media-channel-node-edge-point-spec": {
                                    "mc-pool": {
                                        "occupied-spectrum": [
                                            {
                                                "frequency-constraint": {
                                                    "adjustment-granularity": "G_50GHZ",
                                                    "grid-type": "DWDM"
                                                },
                                                "lower-frequency": 195475000,
                                                "upper-frequency": 195525000
                                            }
                                        ],
                                        "supportable-spectrum": [
            …
```

Implementation of Dijkstra

- python code for implementation of Dijkstra algorithm and getting node also routing with short path

jupyter Dijkstra.py

Logout

File Edit View Language Python

```
    def get_id(self):
        return self.id

    def get_weight(self, neighbor):
        return self.adjacent[neighbor]

    def set_distance(self, dist):
        self.distance = dist

    def get_distance(self):
        return self.distance

    def set_previous(self, prev):
        self.previous = prev

    def set_visited(self):
        self.visited = True

    def __str__(self):
        return str(self.id) + ' adjacent: ' + str([x.id for x in self.adjacent])

class Graph:
    def __init__(self):
        self.vert_dict = {}
        self.num_vertices = 0

    def __iter__(self):
        return iter(self.vert_dict.values())

    def add_vertex(self, node):
        self.num_vertices = self.num_vertices + 1
        new_vertex = Vertex(node)
        self.vert_dict[node] = new_vertex
        return new_vertex

    def get_vertex(self, n):
        if n in self.vert_dict:
            return self.vert_dict[n]
        else:
            return None

    def add_edge(self, frm, to, cost = 0):
        if frm not in self.vert_dict:
            self.add_vertex(frm)
        if to not in self.vert_dict:
```

- https://gitlab.com/sdn-nfv-lab-course/2019-2020/sdn-project-5

DIPARTIMENTO DI ELETTRONICA
INFORMAZIONE E BIOINGEGNERIA
POLITECNICO
MILANO 1863

THANKS

Question time