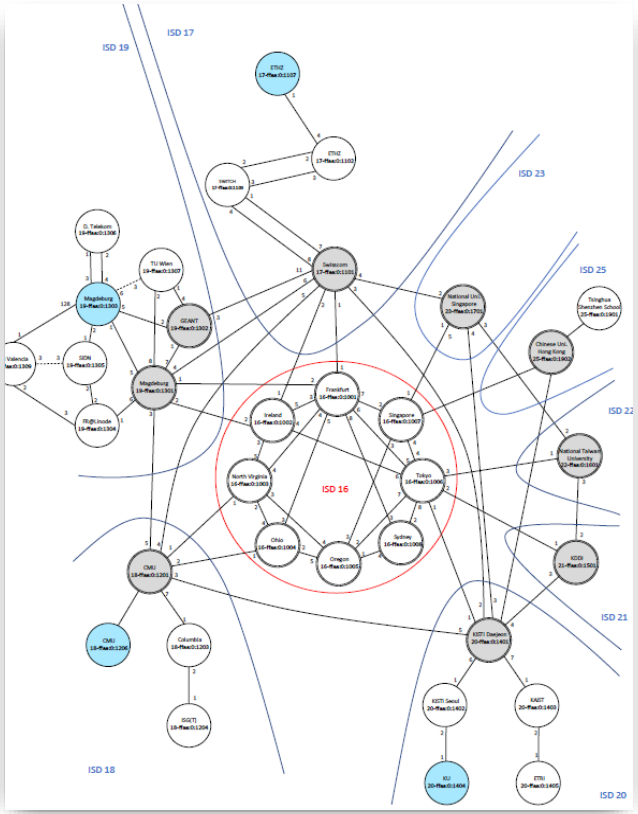
SCION: Heir to the throne ♛

A next-generation **MOBA**

MOBA: Multiplayer Online ~~Battle Arena~~

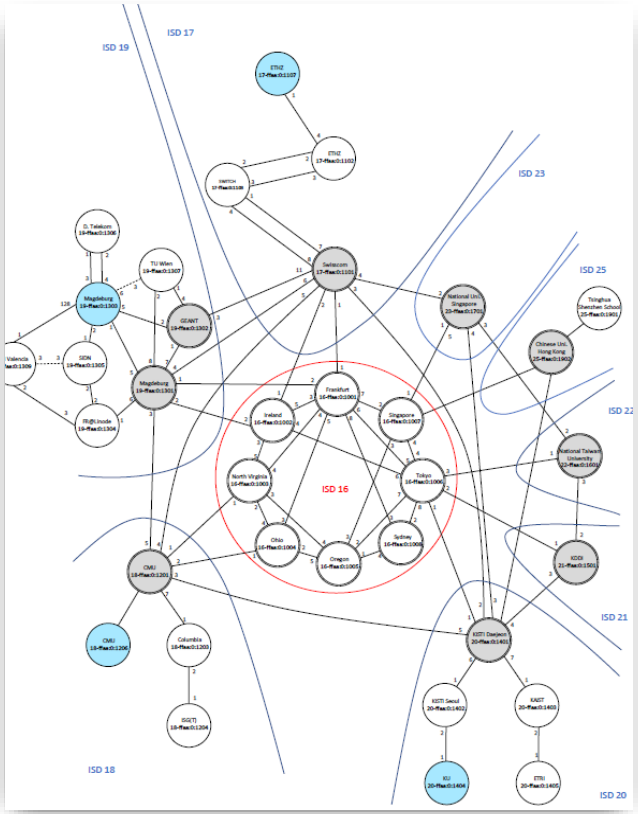
BANDWIDTH ATTACK !!

- Play on the **real SCIONLab network**
- Control endpoints on this topology
- Create a **sender** application to transfer data

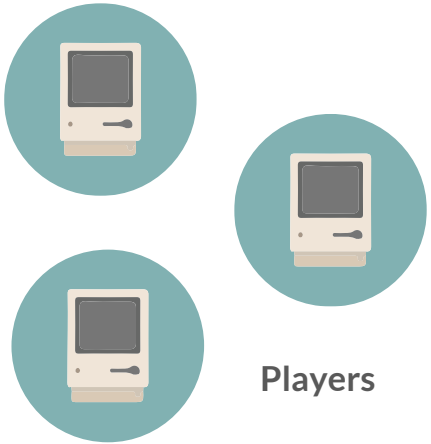


Aim of the Game

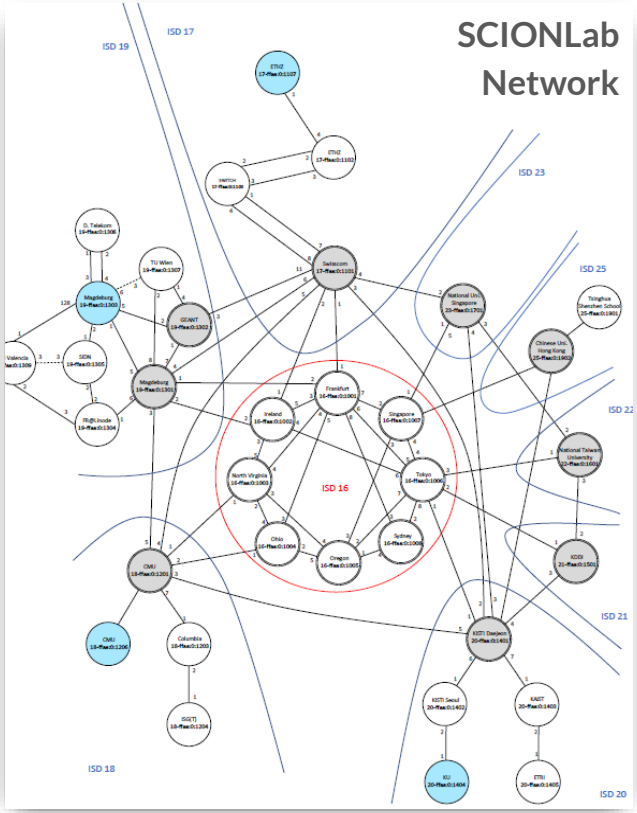
- Create an application that delivers as much data as possible over the network.
- The application will run on a source server…
- … and will send data to pre-defined sink server
- Later who selects sources, sinks etc…



“How to play”



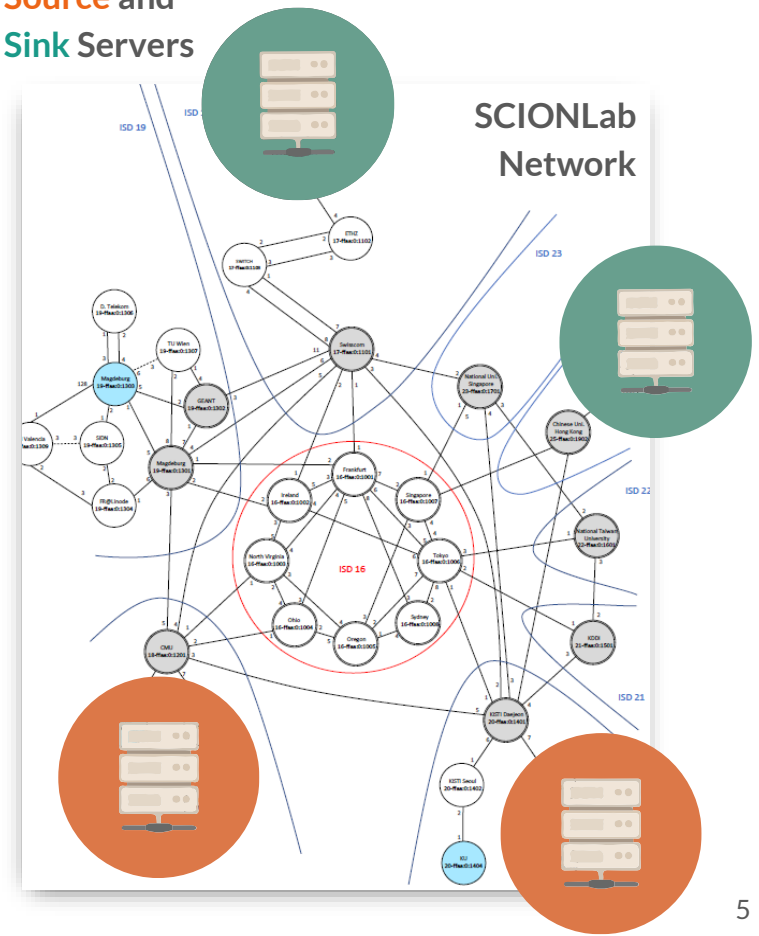
Players



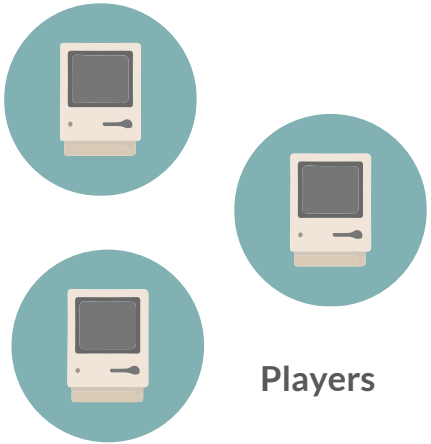
“How to play”

Players

Source and
Sink Servers



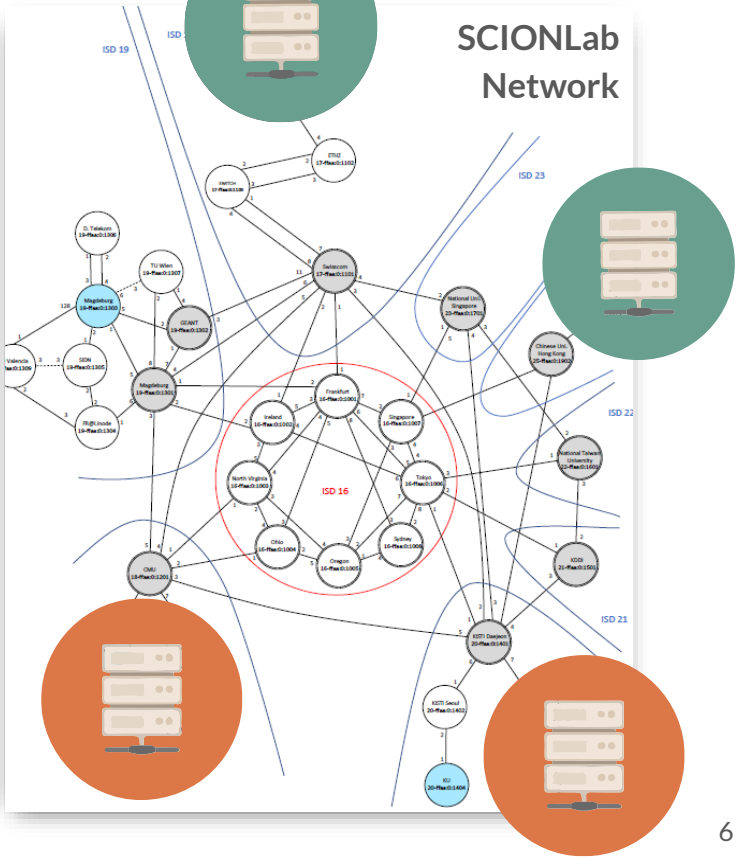
“How to play”



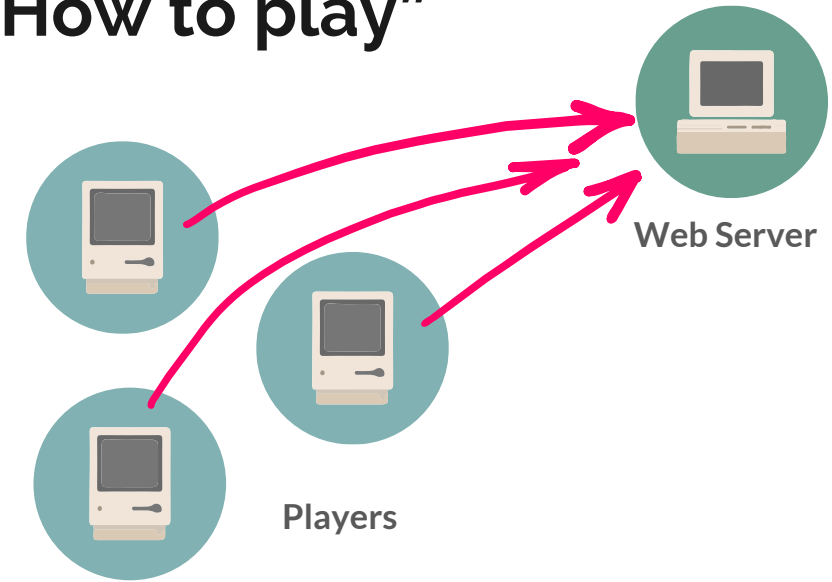
Players

Web Server

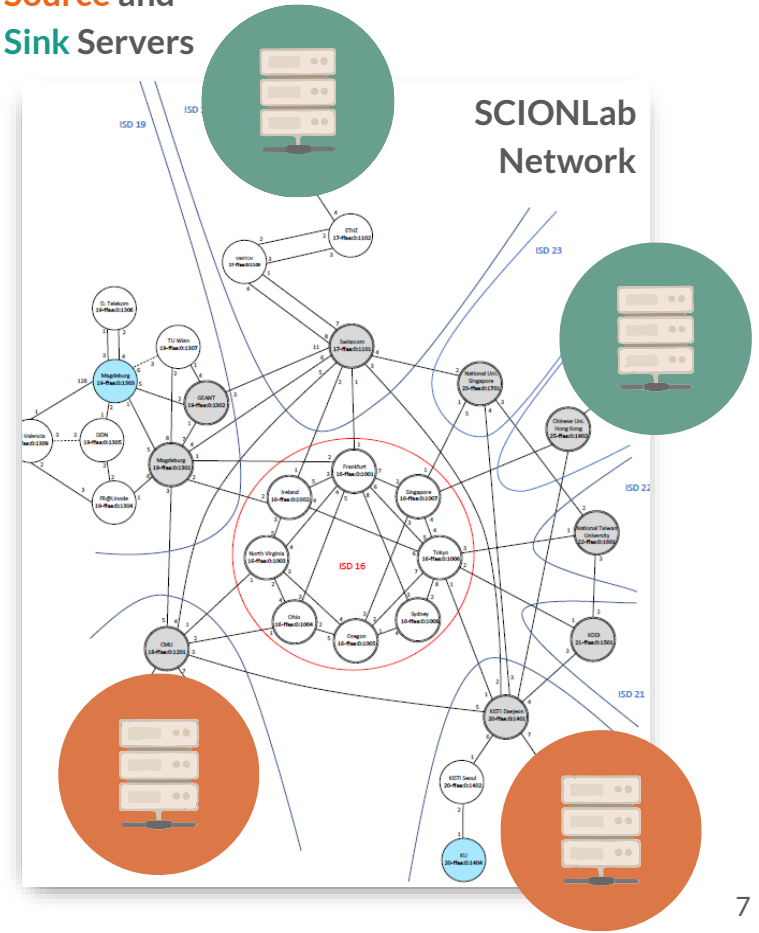
Source and
Sink Servers



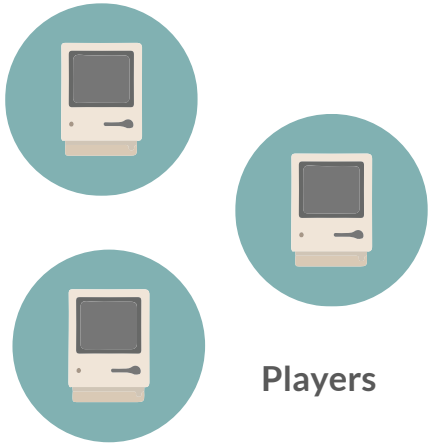
“How to play”



Source and
Sink Servers



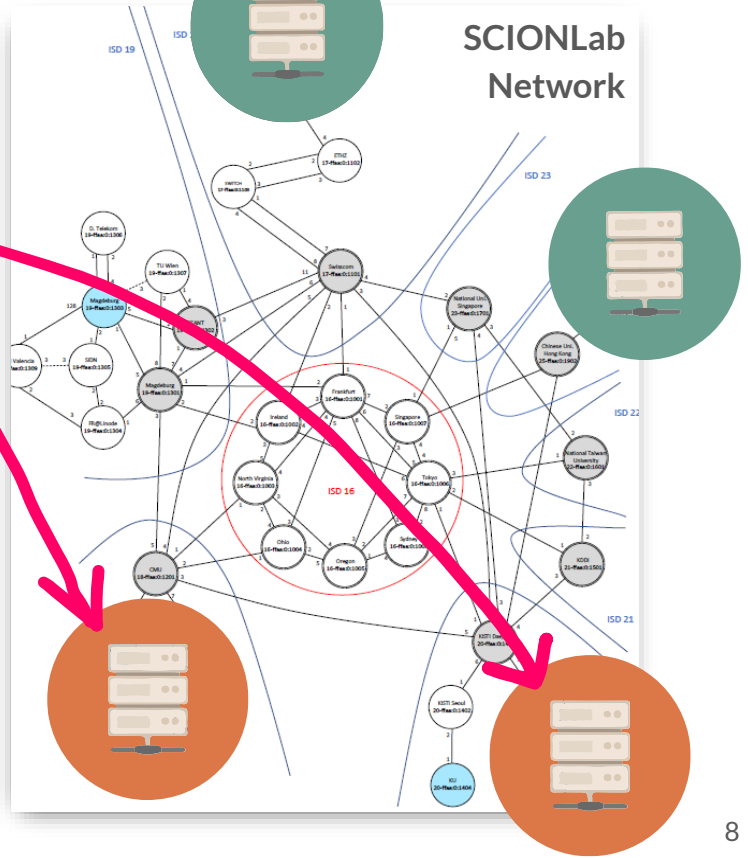
“How to play”



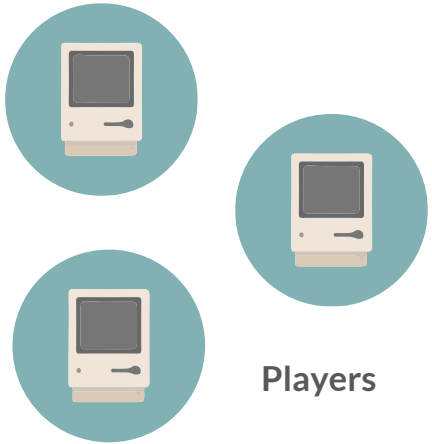
Players

Source and
Sink Servers

Web Server



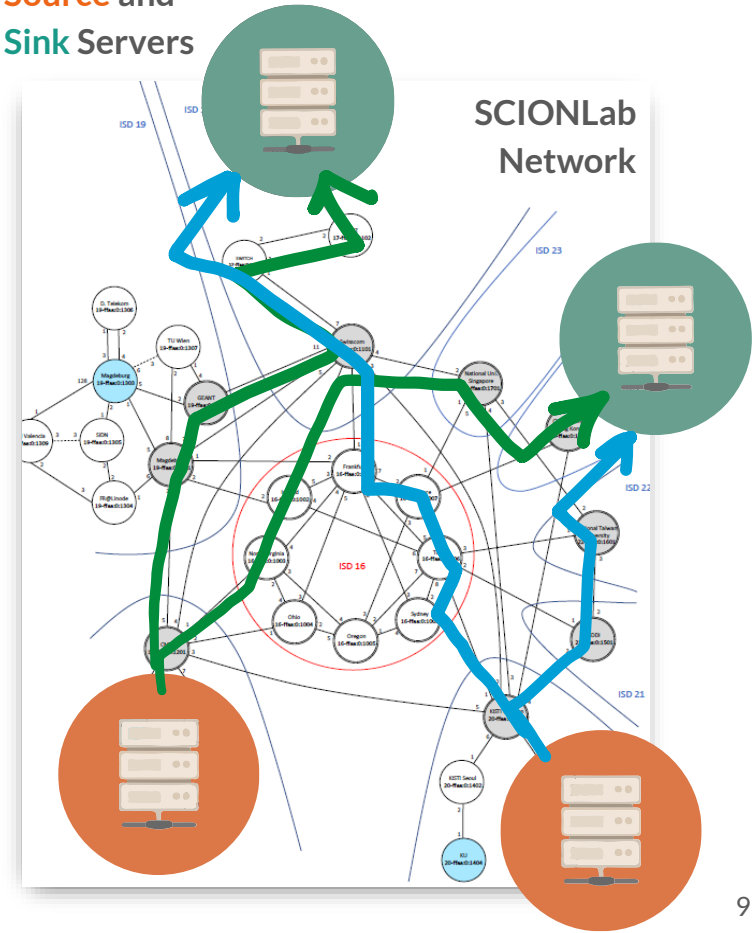
“How to play”



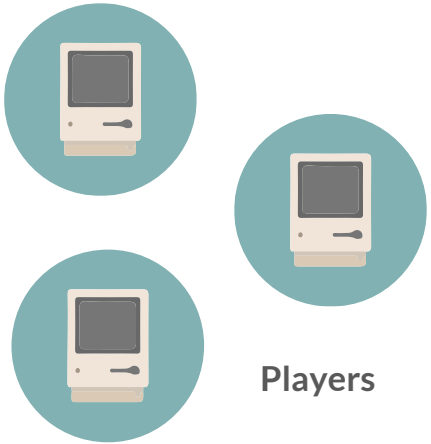
Players

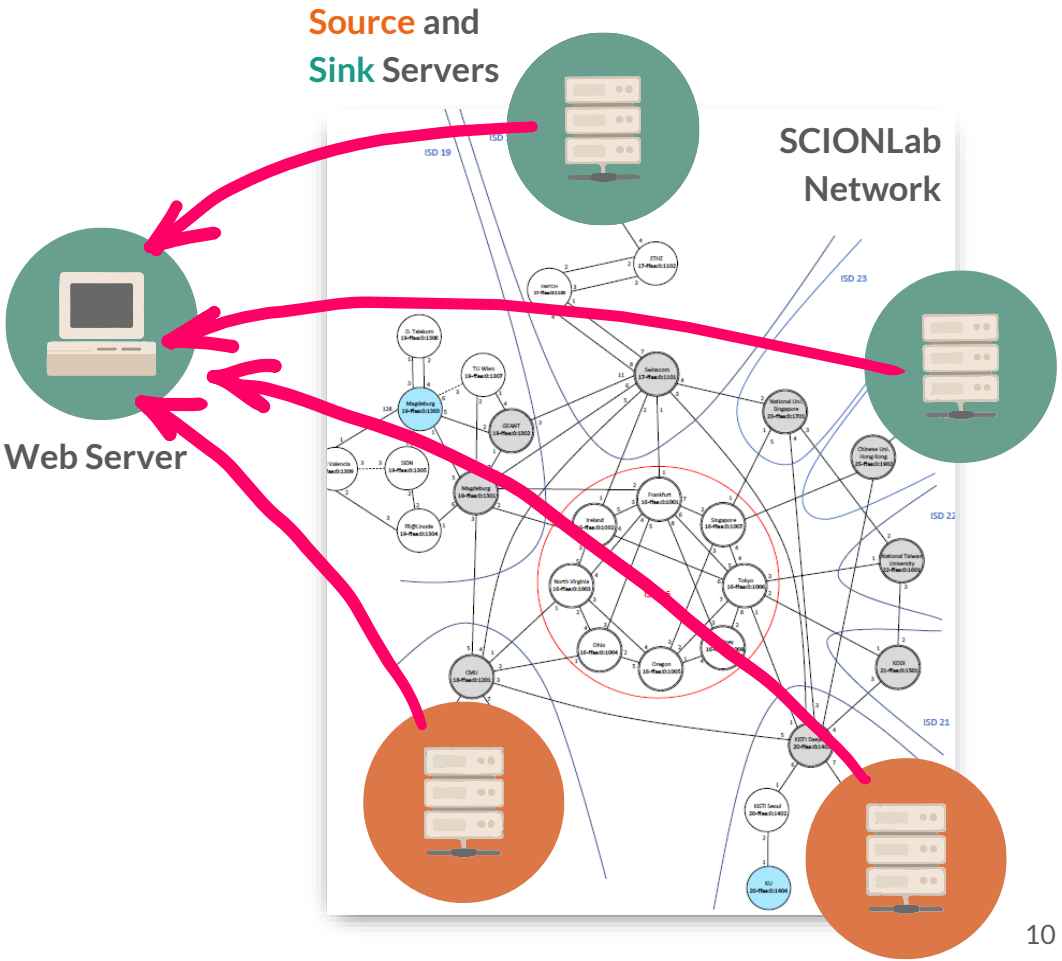
Web Server

Source and
Sink Servers

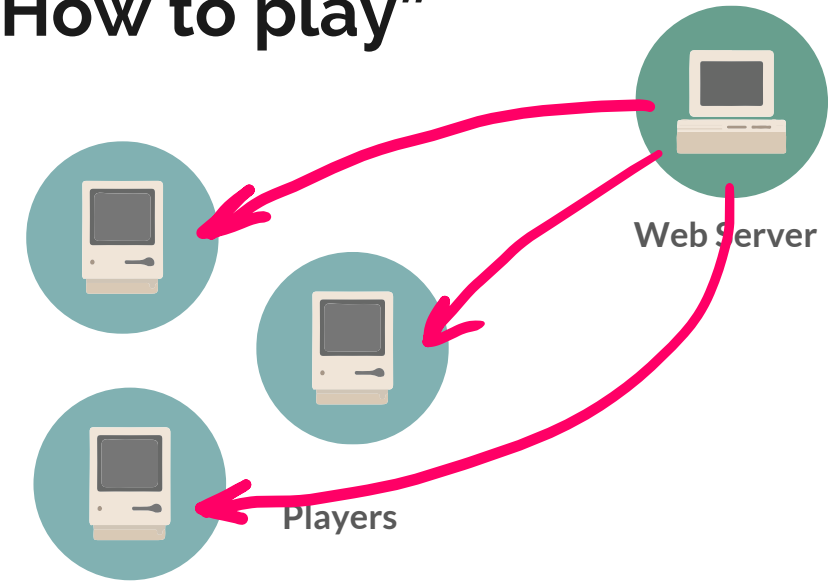


“How to play”

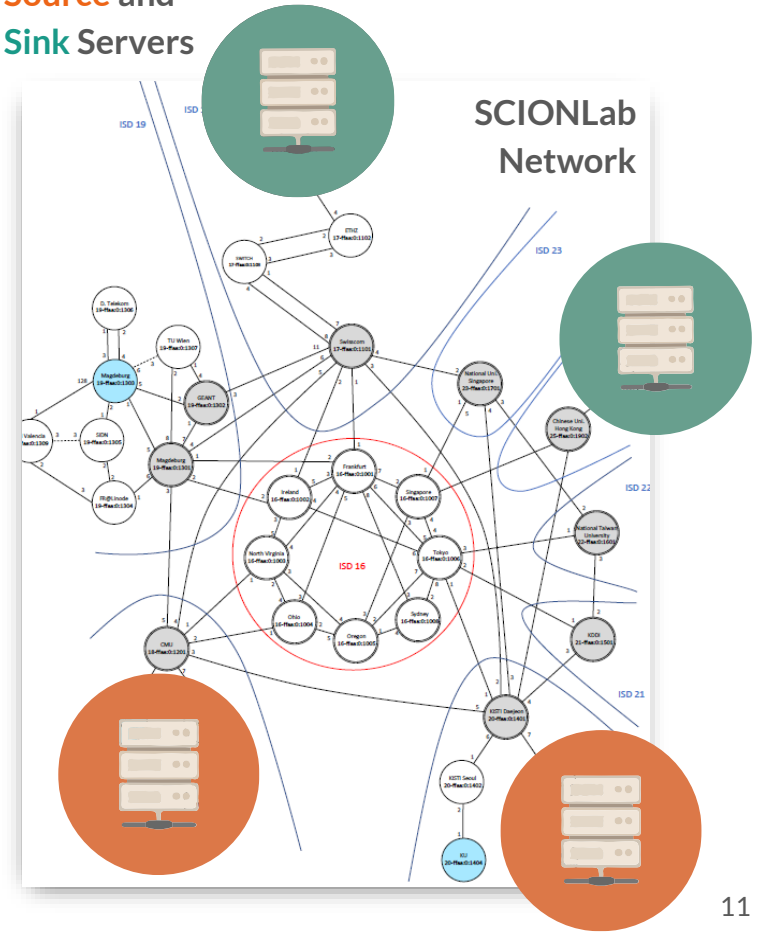

Players


Source and
Sink Servers
SCIONLab
Network
Web Server

“How to play”

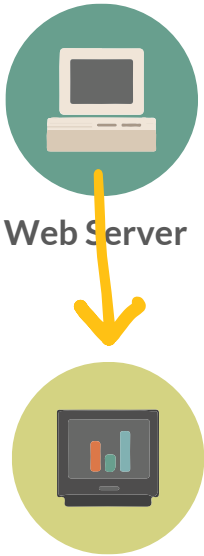


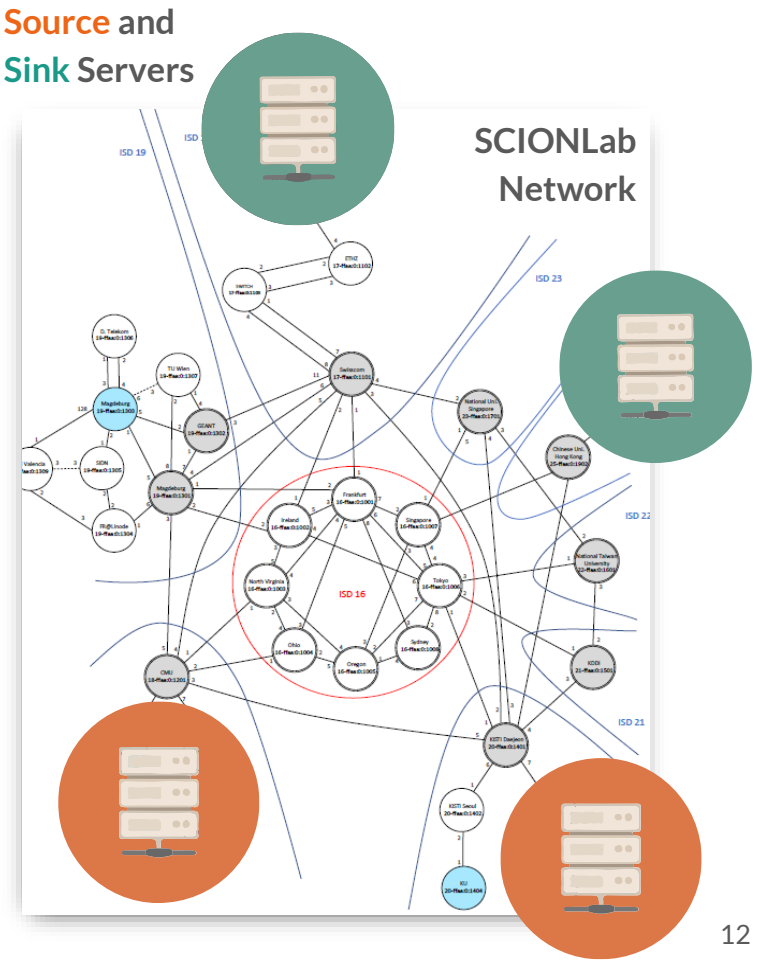
Source and
Sink Servers



“How to play”

Players



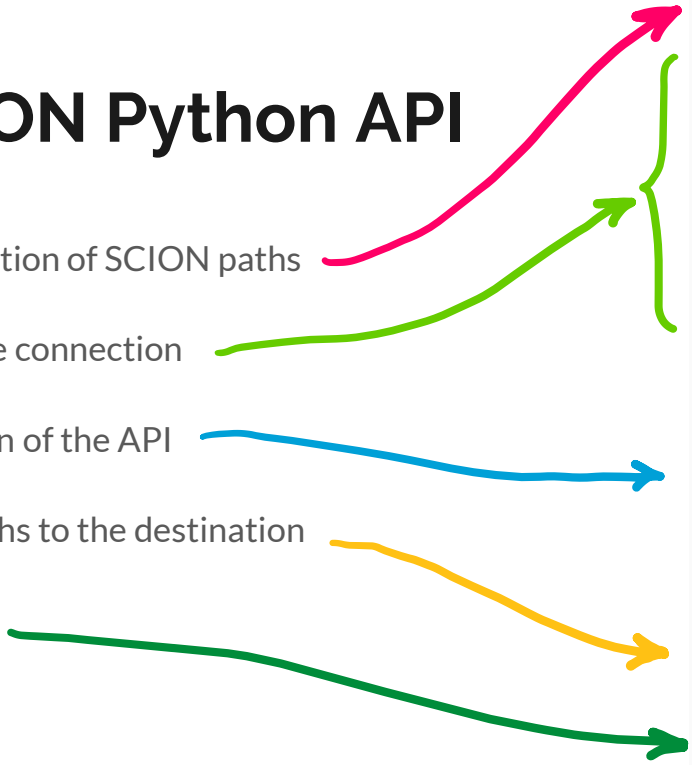


Game Rounds

- The game happens in rounds, one every **90** seconds
- In every round, the latest version is run from the source for **30** seconds
- At the end of the round, the number of transferred bytes is collected and the score is computed
- **1 source** and **3 sinks** are computed randomly at each round
- **10 rounds** are for **“testing”**, **10 rounds** count towards the **final score**!

The SCION Python API

- Representation of SCION paths
- Socket-type connection
- Initialization of the API
- Getting paths to the destination
- Listening



```python
class Path:
    ...

class connect:
    def __init__(self, destination, path):
        ...
    def write(self, buffer):
        ...
    def read(self, buffer):
        ...
    def close(self):
        ...

def set_log_level(level):
    ...

def init():
    ...

def local_address():
    return '1-ffaa:0:0,[127.0.0.1]'

def paths(destination):
    return [Path(), Path()]

def listen(port):
    return connect(None, None)
```

Example code 1

```
9  def main():
10     sci.init()
11     print('Local Address is {}'.format(sci.local_address()))
12     for dest_addr, nbytes in parse_tasks_from_stdin():
13         destination = "{}:12345".format(dest_addr) # send to port 12345
14         print(' === TASK to destination {} : {}MB ==='.format(destination, int(nbytes/1024/1024)))
15         paths = really_get_paths(destination)
16         print('Got %d paths' % len(paths))
17         with sci.connect(destination, paths[0]) as fd:
18             for i in range(int(nbytes / 1000)+1):
19                 fd.write(MTU*b'a')
```

Example code 2

```
22  def really_get_paths(destination):
23      # getting paths is async, so let's just retry until we get some
24      while True:
25          try:
26              return sci.paths(destination)
27          except sci.SCIONException:
28              time.sleep(0.1)
29
30
31  def parse_tasks_from_stdin():
32      def parse_line(line):
33          dest, nbytes = line.split()
34          return dest, int(nbytes)
35      return [parse_line(line) for line in sys.stdin]
```

Getting creative!

- Use multiple paths to the same destination
- Multithreading
- DoS the paths of your adversaries
- … you name it!

SCION
you select path
explicit trust
✓
no convergence
✓
✗

Signing up, submitting, getting the logs

REMEMBER TO `$ export TEAM_TOKEN=<token>`

Signup your team	`./scionlab.sh signup TEAMNAME`
Submit new code	`./scionlab.sh submit FILENAME`
Get the latest logs	`./scionlab.sh log`