Computer Networks
@CS.NCTU

Lab. 3: Admission Control

Instructor: Kate Lin

Deadline: 2021.01.09 23:59

Objectives

- In this lab, we are going to write an ***Offline Scheduler*** and an ***Online Scheduler*** to perform ***Admission Control*** in a simple network

1. Learn what is **Admission Control**
2. Learn differences between **Offline** & **Online Scheduling**
3. Learn how to design **Admission Control Algorithms**

TODO

- **Coding**
  - Implement your scheduler in **Offline_AC.py** and **Online_AC.py** for offline scheduling and online scheduling, respectively
  - Modified from the example codes: ex_Offline_AC.py and ex_Online_AC.py

- **Report**
  - Answer the following questions into **Report.pdf**
    - **How do you design your schedulers**
    - **How do you implement your schedulers** (explained with the screenshots of your code)

Agenda

- Background
- Scenario
- File Structure
- **Task**
- Submission
- Grading Policy

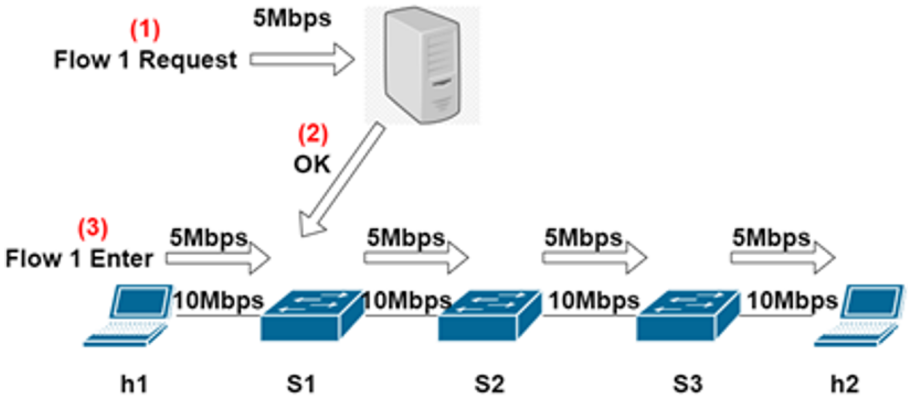
Background

Admission Control

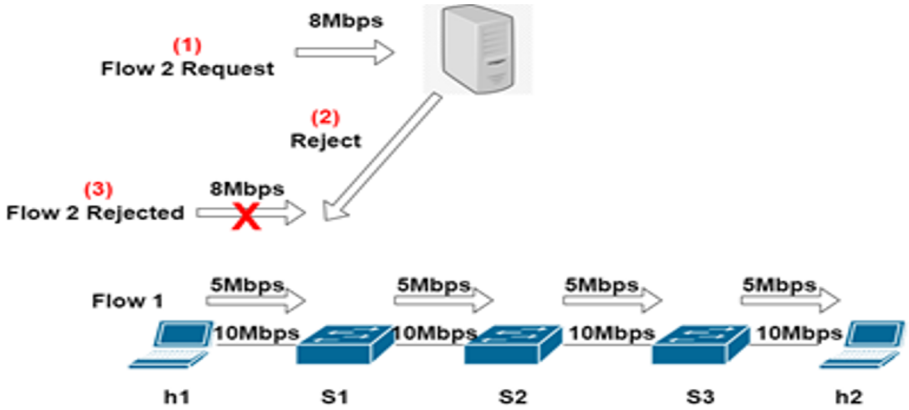
- **Admission control (AC)** is a validation process in communication systems where a check is performed before a connection is established
- A flow should only be permitted if the available resources are sufficient to guarantee QoS
  - e.g., allocated flow rate > requested flow rate

Admission Control (cont.)

- **Resource enough**
  - Flow might be permitted



- **Resource not enough**
  - Flow should be rejected



- AC1 = [Permit, Reject, Reject, Reject]
- AC2 = [Reject, Permit, Permit, Permit]

Goal: Maximize the number of permitted flows

Constraints: QoS of each permitted flow should be satisfied

Hint: permitting the current flow may lead to rejection of more future requests

Offline AC vs Online AC

- Offline AC schedule
  - Global view of flows
  - Schedule all flows at a time
- Online AC schedule
  - Unknown future arrival patterns
  - Schedule each flow request one by one

Offline AC Scheduling

- Input: a sequence of flow requests
  - Each request consists of flow ID and its arrival time
  - e.g., F = [(f1, 1), (f3, 3), (f4, 3), …. (f4, 200)]

- Schedule all the flows at a time
  - e.g., AC = [Permit, Reject, Permit, …, Permit]

- Hint: you can look ahead the future requests
- Hint: do statistics for each small time interval
- Hint: brute-force search

Online AC Scheduling

- Why do we need online decision?
  - Future requests are usually unknown
- How to do on line decision?
  - Assume future arrival patterns are similar to historical patterns
  - Do decision based on historical requests
- Schedule each flow request one by one
  - Historical requests
    1. t = 2-3, f = f1(40%), f2(20%), **f3(20%)**, f4(10%), f5(10%)
    2. t = 2-3, f = f1(40%), f2(20%), f3(20%), **f4(10%)**, f5(10%)
  - Online decision
    1. t = 3, f = f3 ⇒ AC = Serve
    2. t = 3, f = f4 ⇒ AC = Permit

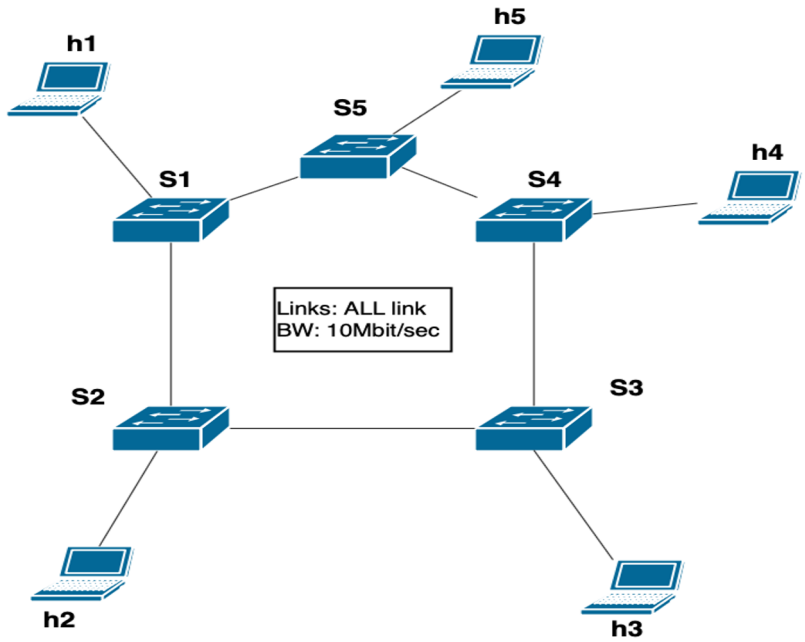
Lab Scenario

Lab3 Scenario

- Network Topology
- Flow Arrival Patterns
- Routing Path
- Evaluation Rule
- AC Scheduling

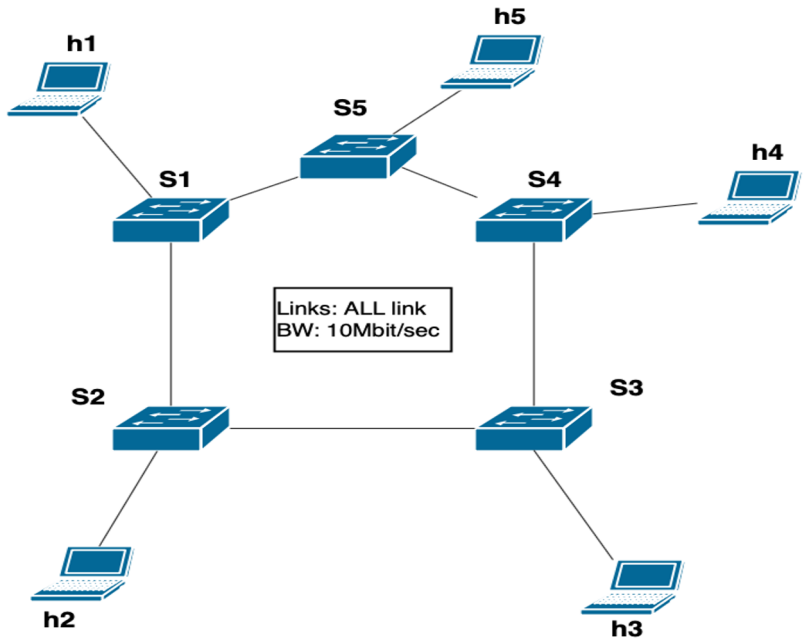
Network Topology

- Each link with same bandwidth of **10** Mbps
- Assume uplink and downlink share the bandwidth
  - flow h2 ⇒ h3 enter network
  - residual bw of h2 ⇒ s2 = residual bw of h2 ⇐ s2



Network Topology (cont.)

- Assume uplink and downlink share the bandwidth
  - If flow request h2 ⇒ h3 is permitted
  - Residual bandwidth of both directions (s2⇒ s3 and s3⇒ s2) both becomes (10 – r)



Flow Arrival Patterns

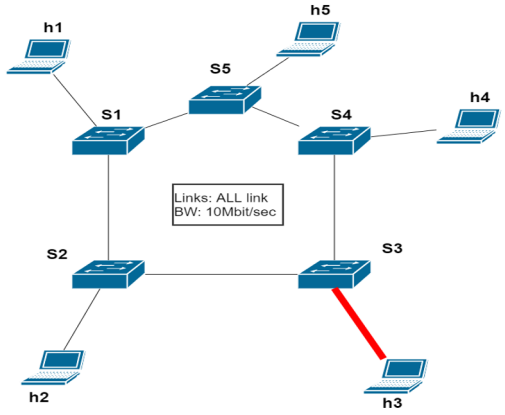
- 5 kinds of flow
  - h1 ⇒ h3, h2 ⇒ h3, h3 ⇒ h4, h4 ⇒ h1, h5 ⇒ h2
- Flow Format
  - src ID, dst ID, arrival time
  - e.g., (1, 3, 1.0), (2, 3, 2.0), ...
- Each flow requests for the same data rate and duration
  - Rate: 5 Mbps
  - Duration: 5 sec
- Arrival Patterns: fixed for the two intervals
  - t = 1:50 (sec): 5 flows [~40%, ~20%, ~20%, ~10%, ~10%]
  - t = 51:100 (sec): 5 flows [~40%, ~20%, ~20%, ~10%, ~10%]

Routing Path

- Fixed shortest path routing
  - f1: h1 ⇒ h3
    - p1: h1□s1□s2 □ s3 □ h3
  - f2: h2 ⇒ h3
    - p2: h2 □ s2 □ s3 □ h3
  - f3: h3 ⇒ h4
    - p3: h3 □ s3 □ s4 □ h4
  - f4: h4 ⇒ h1
    - p4: h4 □ s4 □ s5 □ s1 □ h1
  - f5: h5 ⇒ h2
    - p5: h5 □ s5 □ s1 □ s2 □ h2

Evaluation Rule

- Score basis: 80
  - The basic score is 80 if your code is executable
- **+1**: Each permitted flow that meets its QoS **for the entire duration**
  - Flow rate ≧ rate * 80%
- **-2**: Each permitted flow that does not meet the QoS **in any time interval**
  - Flow rate < rate * 80% in any time interval
- **+0**: rejected flow
- Examples:
  - (h1⇒h3, 0), (h2⇒h3, 1): +1, +1
  - (h1⇒ h3, 0),(h2⇒ h3, 1),(h3⇒ h4, 2): -2, -2, -2
    - # of flow on (h3, S3) = 3
    - 10/3 = 3.33 < 5 * 80% = 4 Mbps



AC Scheduling

- Output your AC decisions
- TAs will call ***evaluate(f,a)*** to measure the bandwidth of each flow and calculate your score accordingly

- NOTE: You **CANNOT** use **global brute-force search**
  - We will manually check it. You will get 0 if you use global brute-force search
  - You may do brute-force search in a small time interval

Sample Code

File Structure

```
lab3-<GITHUB_ID>/                   # This is ./ in this repository
|--- src/                           # Folder of source code
     |--- topo/                     # Folder of topology figure
          |--- topo.png
          |--- topo_matrix.txt      # Adjacency matrix
     |--- ex_Offline_AC.py          # Example of offline code
     |--- ex_Online_AC.py           # Example of online code
     |--- install.sh                # Script to install package
     |--- testcase.txt              # Test Case
     |--- Offline_AC.py             # Your program should be here!
     |--- Online_AC.py              # Your program should be here!
|--- Report.pdf                     # Your report
```

topo_matrix.txt

- **Adjacency matrix**
  - Used in both online and offline AC
  - Undirected graph
  - host 1 ~ 5 mapped to node 1 ~ 5
  - switch 1 ~ 5 mapped to node 6 ~ 10
    - topo[1][6] = 10
    - topo[6][7] = 10
  - Undirected graph
    - topo[1][6] = 10
    - topo[6][1] = 10

```
TA root ~ src topo cat topo_matrix.txt
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 10, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 10, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 10, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10
0, 10, 0, 0, 0, 0, 0, 10, 0, 0, 10
0, 0, 10, 0, 0, 0, 10, 0, 10, 0, 0
0, 0, 0, 10, 0, 0, 0, 10, 0, 10, 0
0, 0, 0, 0, 10, 0, 0, 0, 10, 0, 10
0, 0, 0, 0, 0, 10, 10, 0, 0, 10, 0
```

testcase.txt

- Example Flow Sequence
  - 254 flows
  - Flow Format:
    - Src host ID, Dst host ID, Arrival time
    - Each row is a flow request

```
TA root ~ src cat testcase.txt
1 3 1.0
1 3 4.0
1 3 4.0
1 3 4.0
5 2 4.0
3 4 6.0
3 4 7.0
4 1 7.0
1 3 8.0
2 3 8.0
3 4 8.0
1 3 9.0
1 3 10.0
3 4 11.0
2 3 13.0
```

ex_Offline_AC.py

Input the entire flow sequence to Offline_AC

```
##### [TODO] #####
Offline_AC (flow_sequence, topo)
    return [serve] * len(flow_sequence)
#### End here ####

ex_Offline_AC(topo, testcase.txt):
    # Read entire flow sequence
    flow sequence = read from(testcase.txt)

    # Schedule all flows at a time
    ac = Offline_AC (flow_sequences, topo)

    # Evaluate result of AC
    for f, a in (flow sequence, ac):
                evaluate(f, a)

        print ac_result
```

Evaluation Result

```
TA root ~ src python ex_Offline_AC.py
```

```
##################################################################################
###########  Result of Offline Admission Control Scheduling ##############
##################################################################################

Total score of offline AC is -508
```

- In the example code, TA has implemented a function called **"evaluate (f,ac)"** to **calculate your score based on your AC decisions, *ac*, for a sequence of flows, *f***

ex_Online_AC.py

Only input the current request and historical requests to Online_AC

```
##### [TODO] #####
Online_AC (flow, prev_flows, topo)
    return serve
#### End here ####
ex_Online_AC(topo):
    # Read entire flow sequence
    flow sequence = read from(testcase.txt)
    # Online AC schedule
    prev_flow, ac = [], []
    for idx, flow in flow sequence:
        if idx < 20
                prev_flows += flow
        else
           ac += Online_AC (flow, prev_flows, topo)
           prev_flows += flow
    # Evaluate result of AC
    for f, a in (flow sequence, ac):
                evaluate(f, a)
        print ac_result
```

Evaluation Result

```
TA root ~ src python ex_Online_AC.py
```

```
##################################################################################
##########  Result of Online Admission Control Scheduling ##########
##################################################################################

Total score of Online AC is -468
```

- In the example code, TA has implemented a function called **"evaluate (f,ac)"** to **calculate your score based on your AC decisions, *ac*, for a sequence of flows, *f***

install.sh

- Install required package
  - numpy
    - Matrix-like operation
    - Sampling

```
#! /bin/bash


export LC_ALL=C

pip install numpy==1.16

service openvswitch-switch start
```

Tasks

Tasks

1. Environment Setup

1. Offline Admission Control scheduling
   - Write Offline_AC.py based on ex_Offline_AC.py

2. Online Admission Control scheduling
   - Write Onine_AC.py based on ex_Online_AC.py

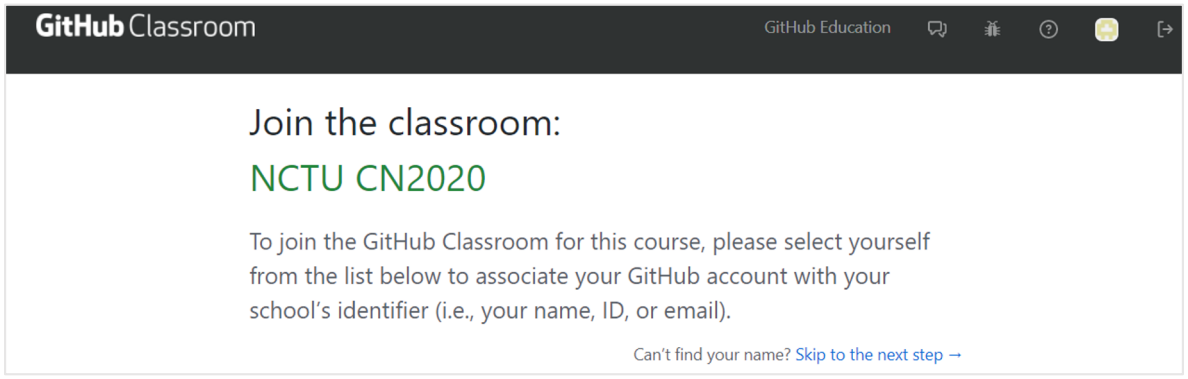
1. Report

Task 1

Environment Setup

Task 1. Environment Setup

- **Step 1. Join the GitHub Classroom Lab3**
  - Click the following link to join this lab
    - GitHub Classroom Lab3
  - Skip to the next step when you see this



  - Go to our GitHub group to see your repository
    - https://github.com/nctucn/lab3-<GITHUB_ID>
  - You will have an initial repository we prepared

Task 1. Environment Setup (cont.)

- **Step 2. Login to your container using SSH**
  - **For windows**
    - Open MobaXterm -> Session -> SSH
      - IP address: `140.113.195.69`
      - Port: port list (Same as Lab.2)
    - Login as root

```
Login: root
Password: cn2019 (Same as Lab.2)
```

  - **For MacOS/Linux**
    - Use terminal to connect to the Docker

```
$ ssh root@140.113.195.69 -p <port>
Password: cn2019 (Same as Lab.2)
```

Task 1. Environment Setup (cont.)

- **Step 3. Change the password (in container)**
  - Change the password of container instead of using the default one

```
$ passwd
Enter new UNIX password:
Retype new UNIX password:
```

  - You will see the following message if succeeded

```
passwd: password updated successfully
```

  - Please remember your own password

Task 1. Environment Setup (cont.)

- **Step 4. Get GitHub repository (in container)**
  - Download required files from GitHub

```
$ git clone https://github.com/nctucn/lab3-
<GITHUB_ID>.git
```

  - Get and set repository for global options

```
$ cd lab3-<GITHUB_ID>/
$ git config --global user.name "<NAME>"
$ git config --global user.email "<EMAIL>"
```

Task 1. Environment Setup (cont.)

- **Step 5. Run Mininet for testing**
  - After logging to your container, you may meet the following error when running Mininet

```
# Run Mininet for testing
$ [sudo] mn
......
*** Error connecting to ovs-db with ovs-vsctl
Make sure that Open vSwitch is installed, that ovsdb-
server is running, and that
"ovs-vsctl show" works correctly.
You may wish to try "service openvswitch-switch start".
```

  - **Solution**

```
# Start the service of Open vSwitch
$ [sudo] service openvswitch-switch start
# Run Mininet again!
$ [sudo] mn
```

Task 1. Environment Setup (cont.)

- **Step 6. Install required package**
  - Run the script below to install required package

```
# Change the directory into /root/lab3-<GITHUB_ID>/src/
$ cd /root/lab3-<GITHUB_ID>/src/
# Run script to install package
$ ./install.sh
```

Task 2

Offline Admission Control Scheduling

Task 2. Offline AC Scheduling

- Design & implement AC algorithm
  - ***Only allowed*** to modify [TODO] section in ex_Offline_AC.py
    - *Only* Offline_AC()

```
###################### [TODO] ###########################
#   Modify this part to create offline Admission Control algorithm
#   Make decision for all flows at a time
#   Decision:
#       1 -> serve flow, 0 -> reject flow
#   Return [decisions]
#   ex:
#       return [1, 1, 1, 1, ...]
#########################################################
def Offline_AC(flow_sequence, topo):
  return [1]*len(flow_sequence)

  ##  Random decision
  #return np.random.randint(2, size=len(flow_sequence)).tolist()

###################### End here #########################
```

Task 2. Offline AC Scheduling

- Testing your Offline AC
  - We will use different testcase.txt to evaluate how good is your algorithm
    - testcase.txt will include
      - Different flow sequence lengths
        - e.g., 254 flows -> 1000 flows
      - Different flow distributions
        - e.g., 5 flows: [~20%, ~20%, ~20%, ~20%, ~20%]

Task 3

Online Admission Control Scheduling

Task 3. Online AC Scheduling

- Design & implement AC algorithm
  - ***Only allowed*** to modify [TODO] section in ex_Online_AC.py
    - *Only* Online_AC()

```
###################### [TODO] ############################
#   Modify this part to create online Admission Control algorithm
#   Make decision for each flow one by one
#   Decision:
#       1 -> serve flow, 0 -> reject flow
#   Return decision
#   ex:
#       return 1
##########################################################
def Online_AC(flow, prev_flows, topo):
  return 1

  # Random decision
  #return np.random.randint(2, size=1)[0]

################### End here #############################
```

Task 2. Online AC Scheduling

- Testing your Online AC
  - We will use different testcase.txt to evaluate how good is your algorithm
    - testcase.txt will include
      - Different flow sequence lengths
        - e.g., 254 flows -> 1000 flows
      - Different flow distributions
        - e.g., 5 flows: [~20%, ~20%, ~20%, ~20%, ~20%]

Task 4. Report

- [TODO] **Your Report.pdf must include**
  - How do you design your AC algorithms
  - How do you implement your AC algorithms (with screenshot)

- You can write your report in English or Chinese

File Structure

```
lab3-<GITHUB_ID>/                   # This is ./ in this repository
|--- src/                           # Folder of source code
     |--- topo/                     # Folder of topology figure
          |--- topo.png
          |--- topo_matrix.txt      # Adjacency matrix of topology
     |--- ex_Offline_AC.py          # Example of offline code
     |--- ex_Online_AC.py           # Example of online code
     |--- install.sh                # Script to install package
     |--- testcase.txt              # Test Case
     |--- Offline_AC.py             # Your program should be here!
     |--- Online_AC.py              # Your program should be here!
|--- Report.pdf                     # Your report
```

Submission

- **Submit your works to your GitHub repository**

```
# In container folder: lab3-<GITHUB_ID>/
# Add files into staging area
$ git add <file>
# Commit your files
$ git commit -m "YOUR OWN COMMIT MESSAGE"
# Push your files to remote
$ git push origin master
```

Submission

- **Push your works to GitHub repository (nctucn)**
  - **AC code** (./src)
    - Offline_AC.py
    - Online_AC.py
  - **Report** (./src)
    - Report.pdf
- **No need to submit to new E3**

Grading Policy

- **Deadline – Jan. 09, 2021. 23:59**
- **Grade**
  - **Offline AC program result – 40%**
    - **feasibility 80% , performance 20%**
  - **Online AC program result – 40%**
    - **feasibility 80% , performance 20%**
  - **Report – 20%**
- Late Policy
  - (Your score) * 0.8^{D}, where D is the number of days over due
- Cheating Policy
  - Academic integrity: Homework must be your own – cheaters share the score
  - Both the cheaters and the students who aided the cheater equally share the score