February 8th, 2017
Exam of Switching technologies for data centers (2016/17)

Rules for the exam. It is **forbidden** to use notes, books or calculators. Use only draft paper provided by the professor. When needed, use approximations.
Time available: 70 minutes.

Problem A

Consider a Top-of-Rack switch with 42 ports, in which 4 ports (denoted as "gold" ports) are adopted to connect to the spine switches and the remaining ports are devoted to the interconnection of the servers within the rack. A QoS policy assures that all the traffic that is directed to any gold port **or** is received by any gold port must be transferred at the highest priority with respect to the remaining traffic.
Assume now that the traffic is only unicast and the switch is implemented through an input queued architecture with Virtual Output Queueing. The scheduler is maximal and implements a strict priority policy on the gold ports.

- Define and describe all the data structures required to define and solve the scheduling problem.
- Describe in pseudocode the scheduling algorithm.
- Discuss the expected performance in terms of throughput and delays, under a generic traffic pattern.

Problem B

Consider the design of a massively large data center based only on Ethernet switches, assuming an oversubscription ratio $4:1$ between server capacity and network capacity. All the adopted switches are equipped with 50 ports running at 1 Gbps. Each server is equipped with one port running at 1 Gbps.

1. For each of the following design problems, draw at high-level the network, compute the required number of switches and the maximum number of supported servers:
   (a) Design the largest possible 2-levels (leaf-and-spine) data center.
   (b) Design the largest possible 2-levels (leaf-and-spine) POD.
   (c) Design the largest possible 3-levels (POD-and-spine) data center.
   (d) Design the largest possible 3-levels (POD-and-spine) POD.
   (e) Design the largest possible 4-levels (POD-and-spine) data center.
2. Discuss how this design is similar to Google's Jupiter data centers.
3. Describe the advantages and disadvantages to adopt a layer-2 addressing and routing scheme if adopted for the above 4-levels data center.

Problem C

Consider the new network paradigm denoted as “Software Defined Networking” (SDN).

1. What is the main difference with the traditional Internet architecture?
2. What is a SDN controller? How can its API interfaces be classified?
3. What is Openflow?
4. How is a flow table defined in Openflow?
5. What are the main messages defined in Openflow and what are their purposes?
6. What are the main consequences of Openflow on the design of the switching architectures?

Hints for the solution

Problem A

```
void scheduler(int **X, int NUM_PORTS, int NUM_GOLD_PORTS) {
    // X is matrix of size NUM_PORTS*NUM_PORTS; X[in][out]=queue length for VOQ[in][out]
    // gold ports are NUM_GOLD_PORTS and correspond to the first NUM_GOLD_PORTS ports of the switch

    int in,out;
    int matching[NUM_PORTS]; // matching[in]=out, otherwise -1
    bool out_reserved[NUM_PORTS],in_reserved[NUM_PORTS];  // boolean reservation vectors

    // init matching and reservation vectors
    for (in=0; in<NUM_PORTS; in++) {
        matching[in]=-1;
        out_reserved[in]=in_reserved[in]=false;
    }
    // Step 1: match gold input ports to any output port
    for (in=0; in<NUM_GOLD_PORTS; in++) {
        for (out=0; out<NUM_PORTS; out++) {
            if (out_reserved[out]) { // skip already reserved outputs
                continue;
            }
            if (X[in][out]>0) {
                // if the VOQ is non empty, store the matching and reserve the inputs and outputs
                matching[in]=out; out_reserved[out]=in_reserved[in]=true;
                break;
            }
        }
    }
    // Step 2: match gold output ports to any input port
    for (out=0; out<NUM_GOLD_PORTS; out++) {
        if (out_reserved[out]) { // skip already reserved outputs
                continue;
        }
        for (in=0; in<IN_PORTS; in++) {
            if (in_reserved[in]) { // skip already reserved inputs
                continue;
            }
            if (X[in][out]>0) {
                // if the VOQ is non empty, store the matching and reserve the inputs and outputs
                matching[in]=out; out_reserved[out]=in_reserved[in]=true;
                break;
            }
        }
    }
    // Step 3: match remaining ports
    for (in=NUM_GOLD_PORTS; in<NUM_PORTS; in++) {
        if (in_reserved[in]) { // skip already reserved inputs
                continue;
        }
        for (out=NUM_GOLD_PORTS; out<NUM_PORTS; out++) {
            if (out_reserved[out]) { // skip already reserved outputs
                continue;
            }
            if (X[in][out]>0) {
                // if the VOQ is non empty, store the matching and reserve the inputs and outputs
                matching[in]=out; out_reserved[out]=in_reserved[in]=true;
                break;
            }
        }
    }
    // now matching contains the desired matching
}
```

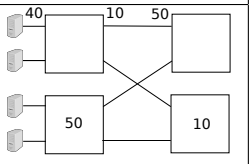
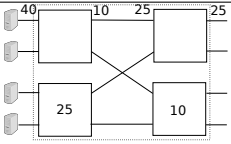
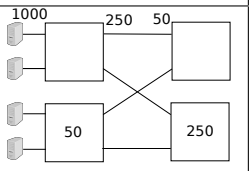
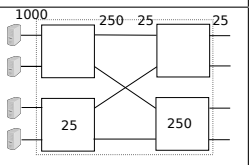
Another possibility for the pseudocode:

```
void scheduler(int **X, int NUM_PORTS, int NUM_GOLD_PORTS) {
    // X is matrix of size NUM_PORTS*NUM_PORTS; X[in][out]=queue length for VOQ[in][out]
    // gold ports are NUM_GOLD_PORTS and correspond to the first NUM_GOLD_PORTS ports of the switch

    int in,out;
    int matching[NUM_PORTS]; // matching[in]=out, otherwise -1
    bool out_reserved[NUM_PORTS],in_reserved[NUM_PORTS];  // boolean reservation vectors

    // init matching and reservation vectors
    for (in=0; in<NUM_PORTS; in++) {
        matching[in]=-1;
        out_reserved[in]=in_reserved[in]=false;
    }
    // Step 1: match gold input or ports
    for (in=0; in<NUM_PORTS; in++) {
        for (out=0; out<NUM_PORTS; out++) {
            // check if the input or the output port is a gold port
            if (in<NUM_GOLD_PORTS OR out<NUM_GOLD_PORTS) {
                // check if the VOQ is not empty the the output is available
                 if (X[in][out]>0 AND !out_reserved[out]) {
                      // store the matching and reserve both the inputs and outputs
                      matching[in]=out; out_reserved[out]=in_reserved[in]=true;
                      break; // consider a new input
                }
        }
    }
    // Step 2: match all the remaining ports
    for (in=0; in<NUM_PORTS; in++) {
        if (in_reserved[in]) { // skip already reserved input
                continue;
        }
        for (out=0; out<NUM_PORTS; out++) {
                // check if the VOQ is not empty the output is available
                 if (X[in][out]>0 AND !out_reserved[out]) {
                        // store the matching and reserve both the inputs and outputs
                        matching[in]=out; out_reserved[out]=in_reserved[in]=true;
                        break; // consider a new input
                }
        }
    }
    // now matching contains the desired matching
}
```

Problem B

Network	Servers	Spine ports	Switches	Layout
2-levels data center	$40 \times 50 = 2,000$	-	$50 + 10 = 60$	
2-levels POD	$40 \times 25 = 1,000$	$25 \times 10 = 250$	$10 + 25 = 35$	
3-levels data center	$1000 \times 50 = 50,000$	-	$50 \times 35 + 250 = 2,000$	
3-levels POD	$1000 \times 25 = 25,000$	$25 \times 250 = 6,250$	$25 \times 35 + 250 = 1,125$	
4-levels data center	$25,000 \times 50 = 1,250,000$	-	$50 \times 1,125 + 6250 = 62,500$