Lecture Seven

Process and Task Scheduling in Network Operating Systems

Algorithms, Implementations, and Performance Trade-offs

Dr. Tarfa Yaseen Hamed

Department of Networks

College of Computer Science and Mathematics

University of Mosul

2025

What is a Scheduler in a NOS?

- *A* **scheduler** *in (*NOS) is a kernel component responsible for allocating system resources (**CPU**, **I/O**, **bandwidth**) to tasks (**processes**, **threads**, or **network flows**) according to predefined policies, ensuring optimal **performance**, **fairness**, and real-time **responsiveness** in networked environments.

Why Scheduling Matters in NOS:

1. Core Challenges in NOS Scheduling

- **Resource Contention:**
  - Multiple services (routing, file sharing, authentication) compete for CPU, memory, and I/O.
  - *Example:* A DNS server handling thousands of queries/sec while also managing BGP updates.
- **Latency Sensitivity:**
  - Network tasks (e.g., VoIP, video streaming) demand predictable delays (< 50ms jitter).
  - Failure Case: Poor scheduling → packet drops → choppy Zoom calls.
- **Scalability:**
  - Must handle 10x load spikes (e.g., DDoS attacks, cloud autoscaling).

Why Scheduling Matters in NOS: (cont.)

2. Consequences of Poor Scheduling

Issue	Impact	Real-World Example
CPU Starvation	Critical tasks (e.g., routing) get delayed	Router drops OSPF updates
I/O Bottlenecks	Disk/network queues overflow	NFS file server becomes unresponsive.
Priority Inversion	Low-priority tasks block high-priority ones	FTP transfer slows down VoIP on a firewall.

Why Scheduling Matters in NOS: (cont.)

3. **Unique NOS Scheduling Requirements**

- **Task Prioritization:**
  - Routing protocols (BGP/OSPF) > User traffic > Logging.
  - **Cisco IOS Example**: priority-queue out on interfaces for VoIP.
- **Determinism:**
  - Real-time guarantees for industrial control systems (TSN – Time-Sensitive Networking).
- **Energy Efficiency:**
  - IoT devices need low-power scheduling.

Types of Scheduling in (NOS)

1. Process/Thread Scheduling *(Traditional CPU Focus)*

- **Purpose:** Allocate CPU time to control-plane tasks (e.g., routing protocols).
- **Key Algorithms:**
  - **Priority-Based**: Critical services (e.g., BGP) get higher CPU shares.
  - **Weighted Round Robin (WRR)**: Guarantees minimum CPU to each service.
- **NOS Example:**
  - Cisco IOS: scheduler allocate (4000μs for control plane, 1000μs for interrupts).

Types of Scheduling in (NOS) (cont.)

2. I/O Scheduling *(Disk/Network Bottleneck Management)*

- **Purpose:** Order disk/network operations to minimize latency and maximize throughput. **Methods:**
  - **Deadline Scheduling**: Ensures no I/O request starves (e.g., for storage NOS like NetApp ONTAP).
  - **Anticipatory Scheduling**: is an I/O scheduling algorithm designed to reduce disk seek times.

Types of Scheduling in (NOS) (cont.)

3. Real-Time Packet Scheduling *(Data-Plane Critical)*

- **Purpose:** Guarantee microsecond-level latency for time-sensitive traffic (e.g., VoIP, industrial control) by:
  - **Prioritizing** packets based on deadlines or time-triggered plans.
  - **Enforcing** strict timing guarantees via synchronized clocks (e.g., IEEE 802.1AS).
  - **Minimizing** Head-of-Line (HoL) blocking for critical flows.

Types of Scheduling in (NOS) (cont.)

- **4. Hybrid Scheduling Models *(Modern NOS Trends)***

Combines multiple scheduling algorithms (e.g., priority queues, real-time deadlines, fairness policies) and deployment models (software + hardware) to optimize NOS performance for diverse workloads.

Key drivers include:

- **Workload Diversity:** NOS must handle latency-sensitive (VoIP), bursty (video streaming), and control-plane (routing updates) traffic simultaneously .
- **SDN/Cloud Integration:** Centralized controllers (e.g., OpenDaylight) program schedules dynamically based on global network state .
- **Hardware Limitations:** which includes Application-Specific Integrated Circuit ASIC to handle line-rate scheduling, while CPUs manage complex policies (e.g., QoS hierarchies)

Key Scheduling Algorithms in NOS

1. **First-Come-First-Served (FCFS)**

Principle:

- Tasks are executed in the order they arrive.

Pros:

- Simple to implement (e.g., basic packet buffers).

Cons:

- Head-of-Line (HoL) Blocking: A long task delays short, critical tasks.
- NOS Impact: Unsuitable for latency-sensitive traffic (e.g., VoIP).

Example:

- Early Ethernet switches used FCFS, causing jitter under load.

Key Scheduling Algorithms in NOS (cont.)

2. **Round Robin (RR)**

Principle:

- Cycles through tasks, giving each a fixed time slice ("quantum").

NOS Use Case:

- Fair CPU allocation among control-plane processes (e.g., OSPF, SSH, SNMP).

Limitation:

- Ignores task priority; a low-priority task can delay high-priority ones.

Key Scheduling Algorithms in NOS (cont.)

3. **Priority Scheduling**

Principle:

- Tasks with higher priority always preempt lower-priority ones.

NOS Implementation:

- **Strict Priority:** Cisco IOS prioritizes routing protocols (BGP/OSPF) over user traffic.
- **Risk:** Starvation of low-priority tasks.

Key Scheduling Algorithms in NOS (cont.)

4. Weighted Fair Queuing (WFQ)

Principle:

- Allocates bandwidth proportionally to flow weights.

Math Behind It:

- Each flow gets weight_i / sum(weights) of the bandwidth.

NOS Use Case:

- QoS in SD-WAN (e.g., VoIP gets 50%, HTTP 30%, FTP 20%).

Key Scheduling Algorithms in NOS (cont.)

5. **Deficit Round Robin (DRR)**

Principle:

- Like RR, but accounts for variable packet sizes.
- Each flow gets a "deficit counter" to ensure fairness.

NOS Use Case:

- Juniper’s QoS policies for mixed traffic (small VoIP + large video packets).

Example:

- Flow A (VoIP): 10x 100B packets → 1000B deficit.
- Flow B (Video): 1x 1500B packet → 500B deficit (carries over).

Deficit Round Robin (DRR) Example

- **Scenario Setup**
- **Flows:**
  - **Flow A:** 3 packets of sizes **500B, 600B, 400B**
  - **Flow B:** 1 packet of size **2000B**
- **Quantum (Q): 1500 bytes** (standard MTU size)
- **Deficit Counters (DC):** Start at **0** for both flows
- **Step-by-Step Execution**
- **Round 1 (First Cycle)**
- **Flow A's Turn:**
  - **Add Quantum:** DC_A = 0 + 1500 = 1500
  - **Check Packets:**
    - **Packet 1 (500B):** 500 ≤ 1500 → **Send**, DC_A = 1500 - 500 = 1000
    - **Packet 2 (600B):** 600 ≤ 1000 → **Send**, DC_A = 1000 - 600 = 400
    - **Packet 3 (400B):** 400 ≤ 400 → **Send**, DC_A = 400 - 400 = 0
  - **Result:** All 3 packets sent, DC_A = 0

Deficit Round Robin (DRR) Example (cont.)

- **Flow B's Turn:**
  - **Add Quantum:** DC_B = 0 + 1500 = 1500
  - **Check Packet:**
    - **Packet 1 (2000B):** 2000 > 1500 → **Cannot send**, deficit too low
  - **Result:** No packets sent, DC_B = 1500 (carries over to next round)
- **Round 2 (Second Cycle)**
- **Flow A's Turn:**
  - **Add Quantum:** DC_A = 0 + 1500 = 1500
  - **Check Packets:**
    - **No packets left** (all sent in Round 1)
  - **Result:** DC_A = 1500 (unused, but no packets to send)

Deficit Round Robin (DRR) Example (cont.)

- **Flow B's Turn:**
  - **Add Quantum:** DC_B = 1500 + 1500 = 3000
  - **Check Packet:**
    - **Packet 1 (2000B):** 2000 ≤ 3000 → **Send**, DC_B = 3000 - 2000 = 1000
  - **Result:** Packet sent, DC_B = 1000 (carries over)

Key Scheduling Algorithms in NOS (cont.)

6. Real-Time Algorithms

a. Rate-Monotonic (RM):

- Static priority: Tasks with shorter periods get higher priority.
- *NOS Use:* Industrial NOS (e.g., TSN for factory automation).

b. Earliest Deadline First (EDF):

- Dynamic priority: Task closest to deadline runs first.
- *NOS Use:* 5G URLLC (Ultra-Reliable Low-Latency Communications).

Trade-off:

- RM simpler but underutilizes resources; EDF more complex but optimal.

Key Scheduling Algorithms in NOS (cont.)

7. **Hierarchical Token Bucket (HTB)**

Principle:

HTB is a **hierarchical, class-based** scheduling algorithm that:

- **Guarantees** minimum bandwidth to each traffic class.
- **Allows borrowing** of unused bandwidth.
- **Enforces hard limits** to prevent starvation.

How HTB Works

Step-by-Step Logic:

- **Token Bucket Mechanism:**
  - Each class gets tokens at its rate (e.g., 10 Mbps = 10,000 tokens/sec).
  - Tokens are consumed when packets are transmitted.
- **Hierarchy Enforcement:**
  - Child classes borrow tokens from parents.
  - Excess tokens cascade up the hierarchy.
- **Priority Handling:**
  - Within a class, packets are scheduled by priority (PRIO) or FIFO.