Scheduling

- In other architectures, buffering and service occur on a per-flow basis.
- That is, there is a buffer for each individual flow and the service of each individual flow is differentiated.
- In this way, it is possible to obtain a very fine differentiation of service.
- However, this presents scalability issues, as the schedulers of a core network node may have to manage several thousands of individual flows.
- There is an upper limit of flows that can be managed on an architecture differentiating the service of individual flows.
Managing service classes instead of individual flows eliminates the scalability issue

The number of service classes is much smaller than the number of flows that a scheduler sustains

For example, 10,000 VoIP flows would fit in one service class inside a class-based scheduler

To the contrary, a per-flow based scheduler would have to instantiate 10,000 buffers and it would have to select among 10,000 flows for the transmission of each packet
Scheduling

- Scheduling can be a complex function

- The amount of resources needed to meet the SLA of all service classes sharing a link depends on
  - The compound TCA of each service class
  - The SLA of each service class
  - The scheduling policy (algorithm)

- Connection admission control can be performed properly only if all these items are known
Scheduling

- Given the compound TCA of a service class on a link’s scheduler, referred to as TCA\textsubscript{i}
- Given the SLA of that service class, referred to as SLA\textsubscript{i}
- The total link’s capacity \( C \) consumed by the service classes sharing that link is a function (indeed complex) of the TCAs and SLAs and of the scheduling policy
  - \( C = f((\text{TCA}_1, \text{SLA}_1), (\text{TCA}_2, \text{SLA}_2), (\text{TCA}_3, \text{SLA}_3), (\text{TCA}_4, \text{SLA}_4), \text{policy}) \)
FIFO scheduler

- The FIFO (First-In First-out) scheduler is the simplest.
- However, it is the less useful to offer differentiated QoS.
- All packets, independently on their service class, are stored in the same buffer and they are served in the same order of their arrivals.
- Clearly, only one SLA can be offered to all flows.
- In order to meet all SLAs concurrently, the most stringent SLA must be guaranteed.
- This is clearly very inefficient.
- As a matter of fact, the FIFO scheduler can be used only for Best-Effort networks.
The strict priority scheduler

- The strict priority (SP) scheduler is simple but effective.

- The SP scheduler, when it has to select the next packet to be served, at first examines the highest priority queue (priority 1).

- If the queue stores at least one packet, a packet is fetched from the queue and served.

- If the priority-1 queue is empty, the scheduler examined the priority-2 queue and, if a packet is present, it is served.

- The priority-3 queue is served only if the other queues are empty.
The strict priority scheduler

- For example, with the strict priority scheduler
  - traffic with stringent QoS requirements can be assigned to the highest priority level
  - Best_Effort traffic can be assigned to the lowest priority level
  - Traffic with intermediate QoS requirements can be served within the second priority level
The strict priority scheduler

- For example, with the strict priority scheduler
  - traffic with stringent QoS requirements can be assigned to the highest priority level
  - Best_Effort traffic can be assigned to the lowest priority level
  - Traffic with intermediate QoS requirements can be served within the second priority level
The General Processor Sharing scheduler

- In the *General Processor Sharing* scheduler (GPS), each service class is assigned a weight, ranging from 0 to 1.

- The sum of weights is 1:
  \[ w_1 + w_2 + w_3 = 1 \]

- The \( i \)th service class receives at least a transmission capacity equal to \( w_i C \), where \( C \) is the link’s capacity.

- If a service class is momentarily silent, spare capacity is available.

- In this case, the spare capacity is distributed among the non-silent service classes, proportionally to their respective weight.
The General Processor Sharing scheduler

- For example, let $w_1 = 0.3$, $w_2 = 0.5$, $w_3 = 0.2$
- If service class 3 is silent, the fraction of link’s capacity received by service class 1 is equal to
  - $0.3 + 0.2 \times 0.3 / (0.3 + 0.5) = 0.375$
- And the fraction of capacity received by service class 2 is equal to
  - $0.5 + 0.2 \times 0.5 / (0.3 + 0.5) = 0.625$
- When service class 3 returns active, the link’s capacity is divided according to the respective weights of service classes
The General Processor Sharing scheduler

- GPS schedulers are referred to also as rate-based schedulers, as they assign explicitly a rate of service to each service class.
- There are several implementations of GPS schedulers, such as the family of weighted fair queueing schedulers.

\[ w_1 \quad \quad w_2 \quad \quad w_3 \]