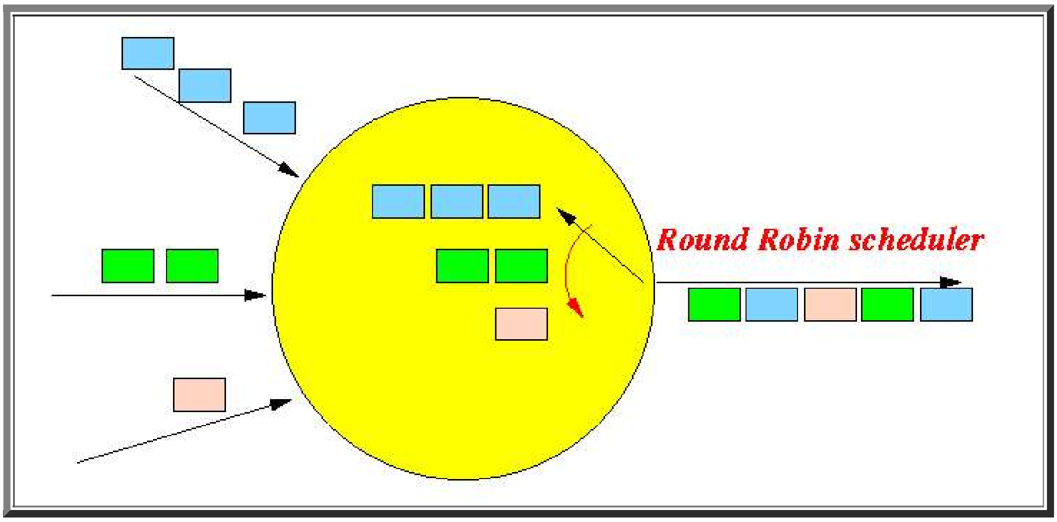
# Deficit Round Robin

- **The Round-Robin scheduler**
  - **Round Robin:**
    - **Packets** from **different flows** are stored in **different queues**
    - **Queues** are **serviced** in a **Round-Robin manner**

    **Graphically:**




- **Property of the Round-Robin scheduler**
  - **Fairness guarantee by Round Robin:**
    - If **flows** $f$ **and** $g$ are **backlogged** over the interval $[t_1, t_2]$ then:
      - $| W_f(t_1, t_2) - W_g(t_1, t_2) | \leq 1$ **packet**
    - **Proof:**
      - Since **flows** $f$ **and** $g$ are **backlogged** over the interval $[t_1, t_2]$, the **Round Robin scheduler** will **transmit one packet from flows** $f$ **and** $g$ in **each round**
      - The **difference** $| W_f(t_1, t_2) - W_g(t_1, t_2) |$ will be **zero**, ***except*** during the time when the **Round Robin scheduler**:
        - has **already** transmitted **a packet from one flow** and
        - **going to** transmit **a packet from the *other* flow**

      ---

      - **Therefore:**
        - $| W_f(t_1, t_2) - W_g(t_1, t_2) | \leq 1$ **packet**

  - **Conclusion:**
    - The **Round Robin scheduler** can achieve **fairness** if
      1. **flows** have **equal weights** (i.e., **unweighted**)
      2. All **packets** have the **same size**

- **Generalizing the Round Robin Scheduling**
  - **Sheerhar** and **Varghese** presented in
    - **"Efficient Fair Queueing using Deficit Round Robin"** ( click here )

    a **very nice generalization** of the **round robin** scheduler that can provide **fairness** when:

    1. **flows** have **different weights**
    2. **packets** can have **different sizes**

---

- **The *unweighted* Deficit Round Robin (DRR) Scheduler**
  - **Information maintained by the *unweighted* DRR scheduler:**
    - **Deficit counter** = the **number of bytes** that a **flow** is **allowed to transmit** when it is its turn.
    - **Quantum** = the **number of bytes** that is **added** to the **deficit counter** of **flow** in **each round**

      ( **Quantum** = **amount of credit per round**)

  - **Operation of the DRR scheduler:**
    - **Packets** from **flows** are **transmitted** in a **round robin** manner
    - The **quantum** is **added** to the **deficit counter** of a flow ***before*** **servicing a flow**.
      - **deficit counter(*f*) = deficit counter(*f*) + quantum**
    - A **packet** from a **flow** is **only transmitted** if:
      - **deficit counter ≥ length of packet**
    - If a **packet** is **transmitted**, the **deficit counter** of that **flow** is **updated** as follows:
      - **deficit counter(*f*) = deficit counter(*f*) − (#bytes in packet)**

D → 0
SE FLOW
VUOTO

**Note:**

- When the **DDR router** detects the **first packet** of a ***new* flow**, it **allocates** a **deficit counter** for the **new flow** and **initialize** the **deficit counter** to **zero**

---

- **Example: *unweighted* Deficit Round Robin Scheduler**
  - **Example:**
    - **3 flows**
    - **Quantum = 500 (bytes)**

    **Graphically:**

```
          Deficit
          counters

          +------+
flow 1    |  0   |
          +------+
          +------+
flow 2    |  0   |
          +------+
          +------+
flow 3    |  0   |
          +------+

Quantum = 500
```

- **Packet arrivals:**

```
          Deficit
          counters

          +------+     +------------+ +---------------------------+
 flow 1   |  0   |     |  200 bytes | |          400 bytes        |
          +------+     +------------+ +---------------------------+
          +------+     +------+ +------------+ +-------------------+
 flow 2   |  0   |     | 100  | |  200       | | 300 bytes         |
          +------+     +------+ +------------+ +-------------------+
          +------+     +------------+ +------------+ +------------+
 flow 3   |  0   |     |  200 bytes | |   200      | |    200     |
          +------+     +------------+ +------------ +------------+

 Quantum = 500
```

- **Servicing Flow 1:**

  - **Add quantum to flow 1's deficiy counter:**

```
          Deficit
          counter

          +------+     +------------+ +---------------------------+
 flow 1   | 500  |     |  200 bytes | |          400 bytes        |
          +------+     +------------+ +---------------------------+
          +------+     +------+ +------------+ +-------------------+
 flow 2   |  0   |     | 100  | |  200       | | 300 bytes         |
          +------+     +------+ +------------+ +-------------------+
          +------+     +------------+ +------------+ +------------+
 flow 3   |  0   |     |  200 bytes | |   200      | |    200     |
          +------+     +------------+ +------------ +------------+

 Quantum = 500
```

  - **Transmit first packet from Flow 1:**

```
          Deficit
          counter

          +------+     +---------------------------+
 flow 1   | 300  |     |          400 bytes        |
          +------+     +---------------------------+
          +------+     +------+ +------------+ +-------------------+
 flow 2   |  0   |     | 100  | |  200       | | 300 bytes         |
          +------+     +------+ +------------+ +-------------------+
          +------+     +------------+ +------------+ +------------+
 flow 3   |  0   |     |  200 bytes | |   200      | |    200     |
          +------+     +------------+ +------------ +------------+

 Quantum = 500
```

**Flow 1** does ***not*** **have enough credits** for the **next packet** and **ends its turn**

**Flow 1** will **carry** its **residual "transmission" credits** over to the **next service round**

- **Servicing Flow 2:**

  - **Add quantum to flow 2's deficiy counter:**

```
          Deficit
          counter

          +------+     +---------------------------+
 flow 1   | 300  |     |          400 bytes        |
          +------+     +---------------------------+
          +------+     +------+ +------------+ +-------------------+
 flow 2   | 500  |     | 100  | |  200       | | 300 bytes         |
          +------+     +------+ +------------+ +-------------------+
          +------+     +------------+ +------------+ +------------+
 flow 3   |  0   |     |  200 bytes | |   200      | |    200     |
          +------+     +------------+ +------------ +------------+

 Quantum = 500
```

  - **Transmit *first* packet from Flow 2:**

```
          Deficit
```

```
           counter

           +------+     +--------------------------+
   flow 1  | 300  |     |          400 bytes       |
           +------+     +--------------------------+
           +------+     +------------+ +-----------------+
   flow 2  | 400  |     |  200       | | 300 bytes       |
           +------+     +------------+ +-----------------+
           +------+     +------------+ +------------+ +------------+
   flow 3  |  0   |     |  200 bytes | |     200    | |     200    |
           +------+     +------------+ +------------  +------------+

   Quantum = 500
```

- **Transmit the second packet from Flow 2:**

```
           Deficit
           counter

           +------+     +--------------------------+
   flow 1  | 300  |     |          400 bytes       |
           +------+     +--------------------------+
           +------+     +-------------------+
   flow 2  | 200  |     |     300 bytes     |
           +------+     +-------------------+
           +------+     +------------+ +------------+ +------------+
   flow 3  |  0   |     |  200 bytes | |     200    | |     200    |
           +------+     +------------+ +------------  +------------+

   Quantum = 500
```

**Flow 2** does ***not* have enough credits** for the **next packet** and **ends its turn**

**Flow 2** will **carry** its **residual "transmission" credits** over to the **next service round**

---

- **Servicing Flow 3:**

  - **Add quantum** to **flow 3's deficiy counter**:

```
           Deficit
           counter

           +------+     +--------------------------+
   flow 1  | 300  |     |          400 bytes       |
           +------+     +--------------------------+
           +------+     +-------------------+
   flow 2  | 200  |     |     300 bytes     |
           +------+     +-------------------+
           +------+     +------------+ +------------+ +------------+
   flow 3  | 500  |     |  200 bytes | |     200    | |     200    |
           +------+     +------------+ +------------  +------------+

   Quantum = 500
```

**Transmit first packet from Flow 3:**

```
           Deficit
           counter

           +------+     +--------------------------+
   flow 1  | 300  |     |          400 bytes       |
           +------+     +--------------------------+
           +------+     +-------------------+
   flow 2  | 200  |     |     300 bytes     |
           +------+     +-------------------+
           +------+     +------------+ +------------+
   flow 3  | 300  |     |     200    | |     200    |
           +------+     +------------  +------------+

   Quantum = 500
```

**Transmit second packet from Flow 3:**

```
           Deficit
           counter

           +------+     +--------------------------+
   flow 1  | 300  |     |          400 bytes       |
           +------+     +--------------------------+
           +------+     +-------------------+
   flow 2  | 200  |     |     300 bytes     |
           +------+     +-------------------+
           +------+     +------------+
   flow 3  | 100  |     |     200    |
```

```
         +------+     +------------+

Quantum = 500
```

**Flow 3** does ***not*** **have enough credits** for the **next packet** and **ends its turn**

**Flow 3** will **carry** its **residual "transmission" credits** over to the **next service round**

---

- **Servicing Flow 1 again:**

  - **Add quantum** to **flow 1's deficiy counter**:

```
        Deficit
        counter

        +------+     +---------------------------+
flow 1  | 800  |     |           400 bytes       |
        +------+     +---------------------------+
        +------+     +-------------------+
flow 2  | 200  |     |     300 bytes     |
        +------+     +-------------------+
        +------+     +------------+
flow 3  | 100  |     |     200    |
        +------+     +------------+

Quantum = 500
```

  - Now **flow 1** has **sufficient credits** to transmit the **next packet** !!!

- And so on....

---

- From the operation, it should be clear that:

  - Flow will share the bandwidth fairly if they are **unsaturated**
  - The **saturated flows** (flows that transmit at a lower rate than the fair share) will receive its complete service rate.

    The left over transmission rate will be divided among the **unsaturated flows**

---

- **Property of the Unweighted DRR scheduler**

  - **Fairness guarantee by DRR:**

    - If the **maximum packet size** $\leq$ **Q** and **flows** ***f*** **and** ***g*** are **backlogged** over the interval **[$t_1$, $t_2$]** then:

      - **$| W_f(t_1, t_2) - W_g(t_1, t_2) | \leq 2 \times Q$ (Quantum)**

    - **Proof:**

      - Suppose there are **k rounds** of **transmissions** in [$t_1$, $t_2$]
      - Since **flows** ***f*** **and** ***g*** are ***continuously*** **backlogged** over the interval **[$t_1$, $t_2$]** and the **maximum packet size < Q**, then:

        - **residual** of **flows** ***f*** < **Q**, and
        - **residual** of **flows** ***g*** < **Q**

      ---

      - The **worst case** is:

        - One flow (e.g., **flow** ***f***) has a **residual** of **Q − 1**
        - The other flow (e.g., **flow** ***g***) is **one round ahead** and has a **residual** of **1**

        ---

        - **Graphically:**

```
                  transmission rounds

             1     2      k-2     k-1     k           # bytes transmitted
---------------------------------------------------    ===================
 flow g:     ..    ..              ..     (1)          k × Q - 1

 flow f:     ..    ..            (Q-1)                 (k-2) × Q + 1
```

- **Therefore:**

  - **$| W_f(t_1, t_2) - W_g(t_1, t_2) | \leq 2 \times Q$ (bytes)**

---

- ***Weighted* Round Robin Scheduling**

  - **Information maintained by the *unweighted* DRR scheduler:**

    - **Deficit counter** = the **number of bytes** that a **flow** is **allowed to transmit** when it is its turn.
    - **Quantum *per flow*** = the **number of bytes** that is **added** to the **deficit counter** of ***that* flow** in **each round**

  - **Operation of the DRR scheduler:**

    - **Packets** from **flows** are **transmitted** in a **round robin** manner
    - The ***flow's* quantum** is **added** to the **flow's deficit counter** of a flow ***before* servicing a flow**:
      - **deficit counter($f$) = deficit counter($f$) + quantum($f$)**
    - A **packet** from a **flow** is **only transmitted** if:
      - **deficit counter ≥ length of packet**
    - If a **packet** is **transmitted**, the **deficit counter** of that **flow** is **updated** as follows:
      - **deficit counter($f$) = deficit counter($f$) − (#bytes in packet)**

  - ***Weighted* DRR:**

    - We can achieve **weighted fairness** by assigning a **different quantum** to **different flows**

  - **Example:**

```
        Flow     Deficit
        Quantum  counter

        +------+ +------+      +------------+ +-------------------------+
 flow 1 | 800  | |  0   |      |  200 bytes | |         400 bytes       |
        +------+ +------+      +------------+ +-------------------------+
        +------+ +------+      +-----+ +------------+ +-------------------+
 flow 2 | 400  | |  0   |      | 100 | |  200       | | 300 bytes         |
        +------+ +------+      +-----+ +------------+ +-------------------+
        +------+ +------+      +------------+ +------------+ +------------+
 flow 3 | 600  | |  0   |      |  200 bytes | |    200     | |    200     |
        +------+ +------+      +------------+ +------------  +------------+
```

  - At the start of servicing a flow, the **flow quantum** is **added** to the **deficit counter**

---

- **Property of the *weighted* DRR scheduler**

  - **Fairness guarantee by *Weighted* DRR:**

    - Let **$Q_f$** = the **quantum** of **flow $f$**
    - If the **maximum packet size of flow $f$ ≤ $Q_f$** and **flows $f$ and $g$** are **backlogged** over the interval **$[t_1, t_2]$** then:

```
| Wf(t1, t2)    Wf(t1, t2) |
| ---------  -  --------- |  ≤ 2
```

```
   |     Q1          Q2     |
```

(Proof is similar to the **unweighted DRR**)

---

---

- **Using DRR Scheduling in NS**
  - The **unweighted DRR scheduler** has been implemented in **NS**
  - To create a **DRR** queue, use:

    ```
    $ns duplex-link $n1 $n2 0.3Mb 200ms DRR
    ```

  - You can set the following parameters of the **DRR** queue:
    - **quantum_** = **size** of the **quantum** that is added per round
    - **buckets_** = **number of buckets**

      **Buckets and flows:**

      - Each **flow** is assigned a **bucket**
      - A **bucket** can contain **multiple flows**

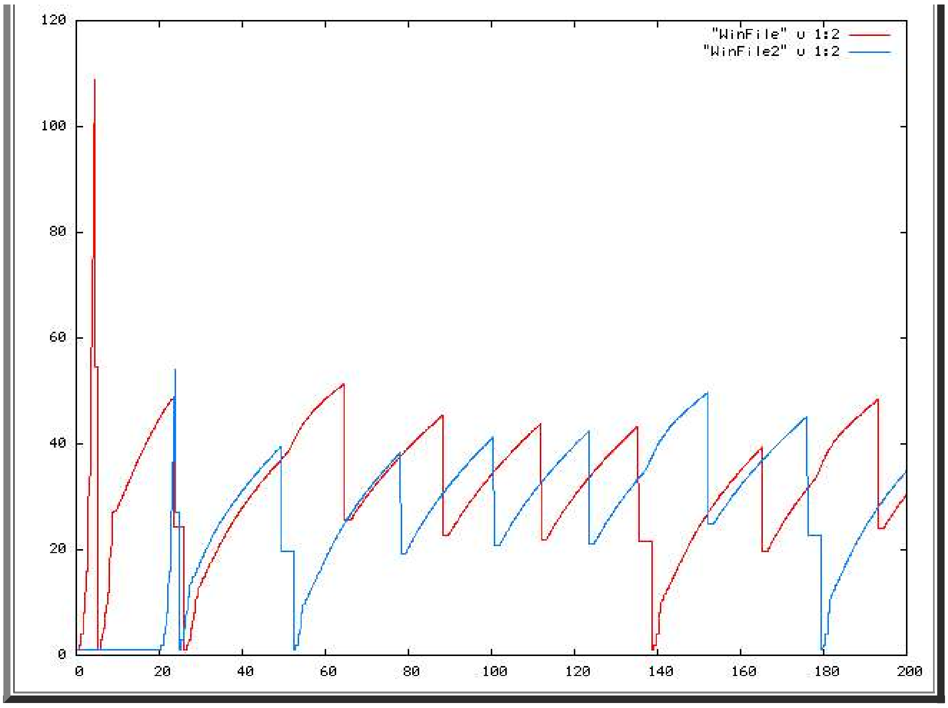
      ---
    - **blimit_** = **bucket limit** ( = the **queue size** in each bucket)

      ---
    - **mask_** = how to **identify a flow** (= how to put flows into buckets).
      - **mask_ = 0**:
        - a **flow** is identified by **(node id, port id)**
        - **In this case**, **each flow** is put in a **different bucket**
      - **mask_ = 1**:
        - a **flow** is identified by **(node id)** (only)
        - **In this case**, ***all* flows** originating from the **same node** are put in the **same bucket**
  - **Example:**

    ```
    Queue/DRR set quantum_  500
    $ns duplex-link $n1 $n2 0.3Mb 200ms DRR
    ```

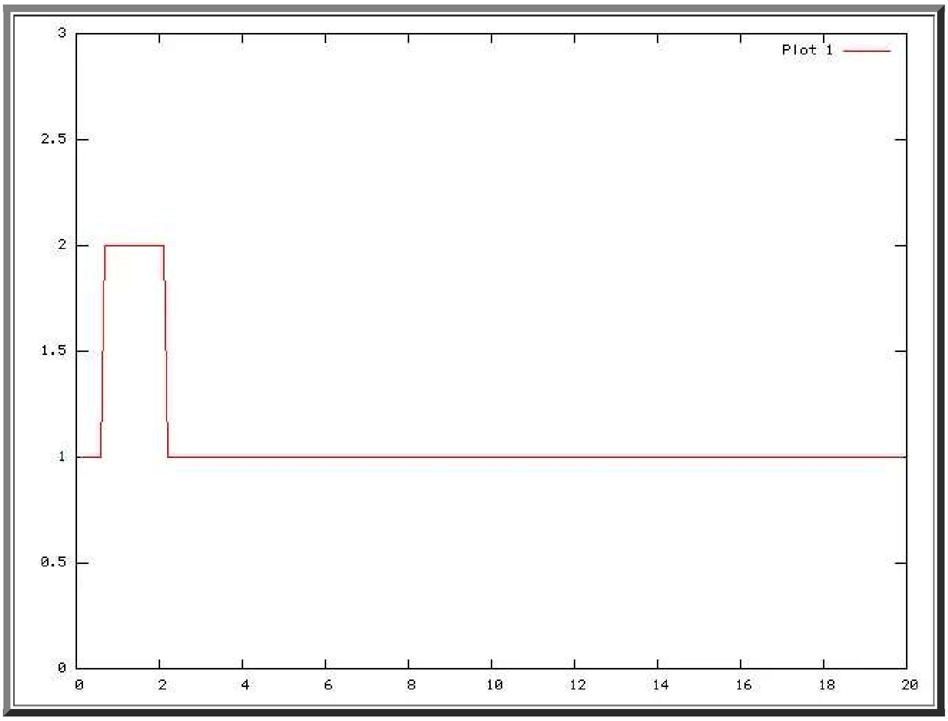
  ---

  - **DEMO:** (2 TCP flows sharing a DRR queue) ***Example***
    - NS Prog file: click here




- **DRR can limit *unresponsive* (UDP) flows**
  - **TCP competing with an *over-sealous* UDP flow using Drop Tail scheduling:**

    **CWND of the TCP flow:**




  - **DEMO:** (1 UDP vs 1 TCP flows)

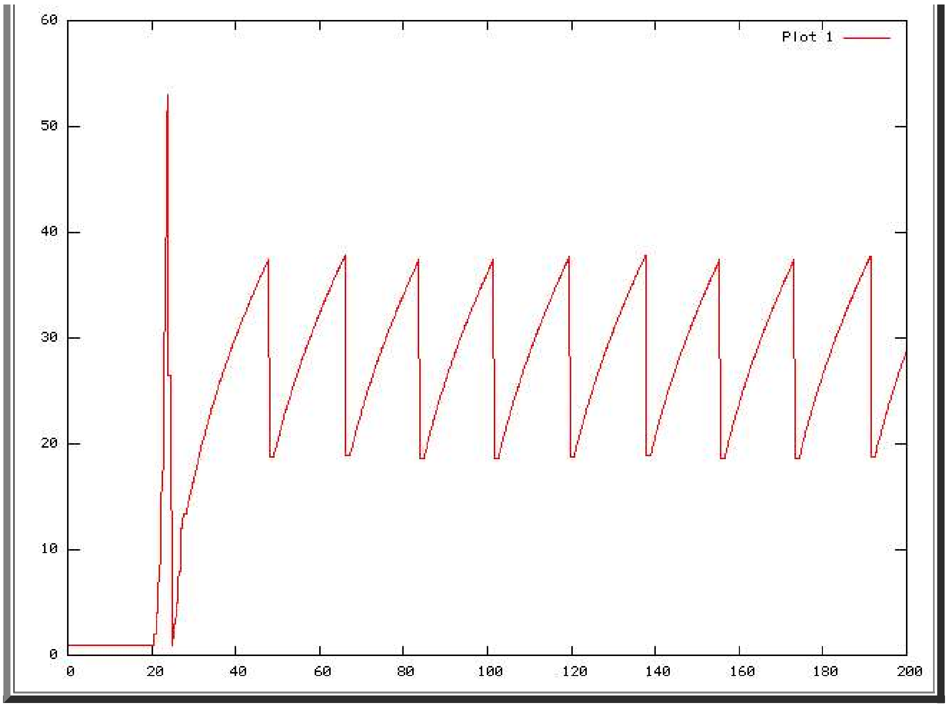

    - NS Prog - UDP and TCP flow sharing DropTail Queue: click here

      The **UDP flow** makes **TCP** use **CWND = 1** all the time...

      (Look at the **NAM** visualization and you will know why...)

  - **TCP competing with an *over-sealous* UDP flow using DRR scheduling:**

    **CWND of the TCP flow:**




---

- **Example Program:** (Demo above code) Example
  - NS Prog - **UDP and TCP flow** sharing **DRR** Queue: click here

**Note:**

- You can see that **DDR** can provide TCP with a **fair share** of bandwidth even when the UDP flow transmits more than the bottle neck link.

---