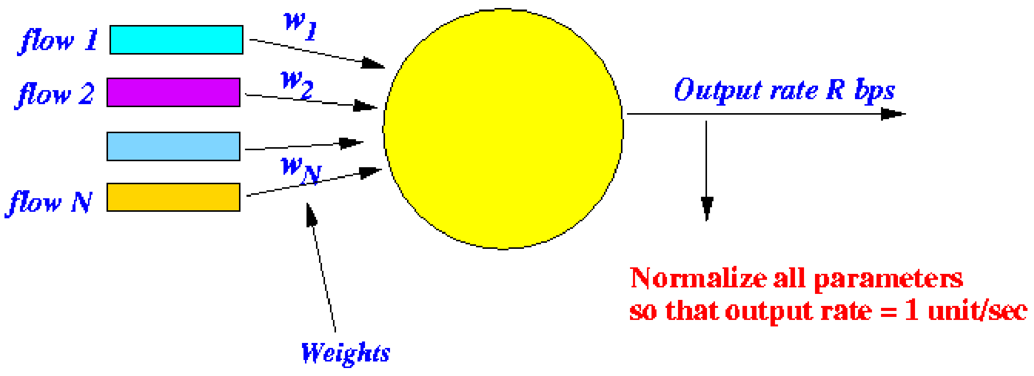
Weighted Fair Queueing: a packetized approximation for FFS/GPS

- **Paper:**
  - A General Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case click here
  - Authors: A. K. Parekh and R. G. Gallager
  - Note: they called their scheme Packet-by-packet GPS (PGPS), but it is now more popularly known as Weighted Fair Queueing (WFQ).
  - Note: their paper also appeared in Infocom93 and received the **best paper** award.

- **System parameters:**



  - ***N*** flows
  - Flow ***i*** has reservation weight w_i

    NOTE: each weight w_i is normalized by ***R***. Hence, all weights are $w_i < 1$

  - The GPS server will guarantee that flow ***i*** will receive **at least** a service rate of

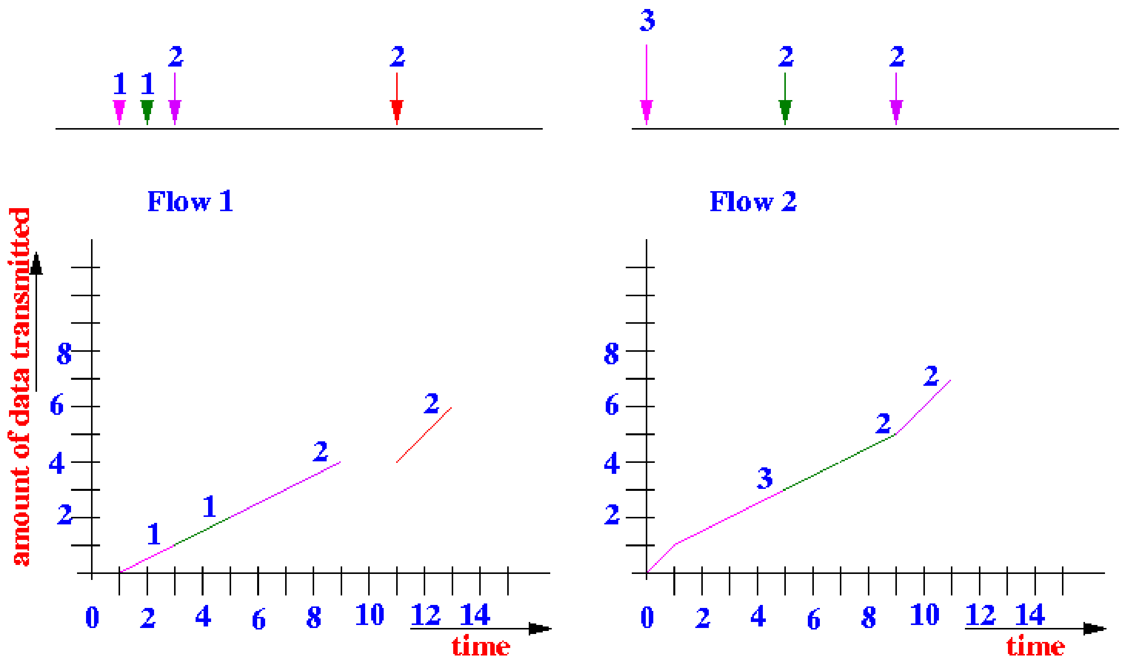
$$g_i = \frac{w_i}{w_1 + w_1 + \dots w_N} R$$

  - Note: I say "at least" because when some other flow ***j*** do not have any packet to send, then flow ***i*** will receive more data rate than the formula above...

- **Example 1 of transmission in GPS**
  - 2 flows
  - Packets arrive "instantaneously" (infinite data rate on input link)
  - Packet arrival times and length:

	Flow 1:		FLow 2:	
	Arrival Time	Packet length	Arrival Time	Packet length
Packet 1:	1	1	0	3
Packet 2:	2	1	5	2
Packet 3:	3	2	9	2
Packet 4:	11	2	-	-

  - Packet length are normalized by transmission rate (i.e., 1 unit of packet length take 1 sec to transmit)
  - The following picture shows how the packets are serviced by a GPS server when $w_1 = w_2 = 0.5$

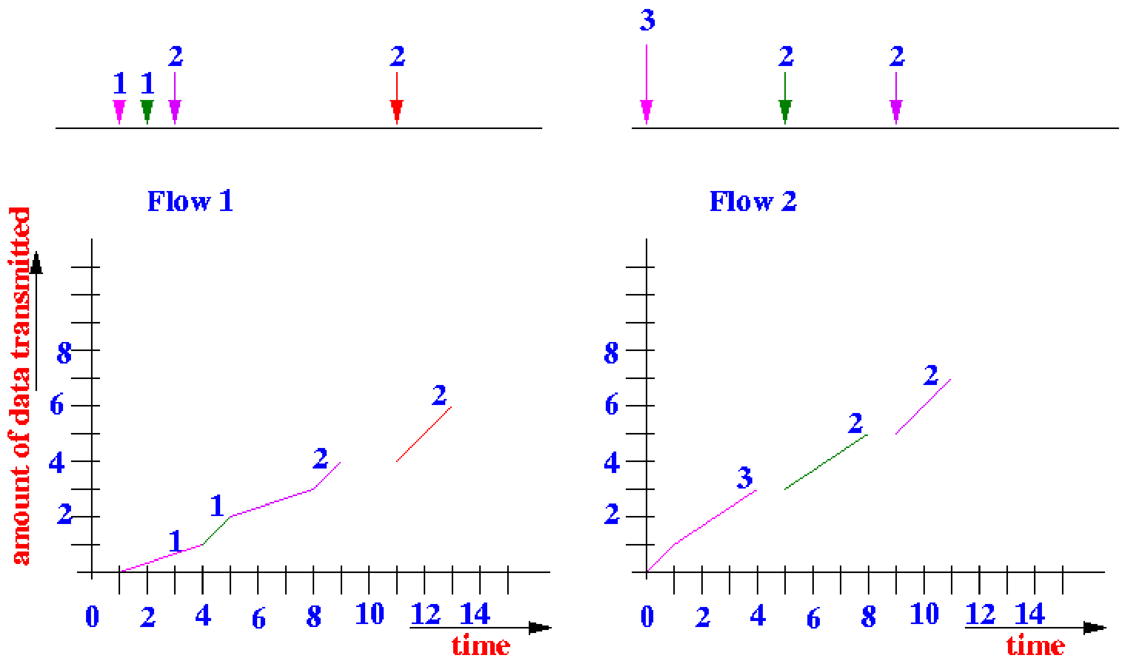


- The arrival and finish times of the packets are as follows:

```
                      Flow 1:                         FLow 2:

             Arrival Time   Finish Time    Arrival Time   Finish Time
Packet 1:         1              3              0              5
Packet 2:         2              5              5              9
Packet 3:         3              9              9             11
Packet 4:        11             13              -              -
```

- **Example 2 of transmission in GPS**
  - The following picture shows how the **same arrival of packets** are serviced by a GPS server when $w_1 = 1/3$ and $w_2 = 2/3$

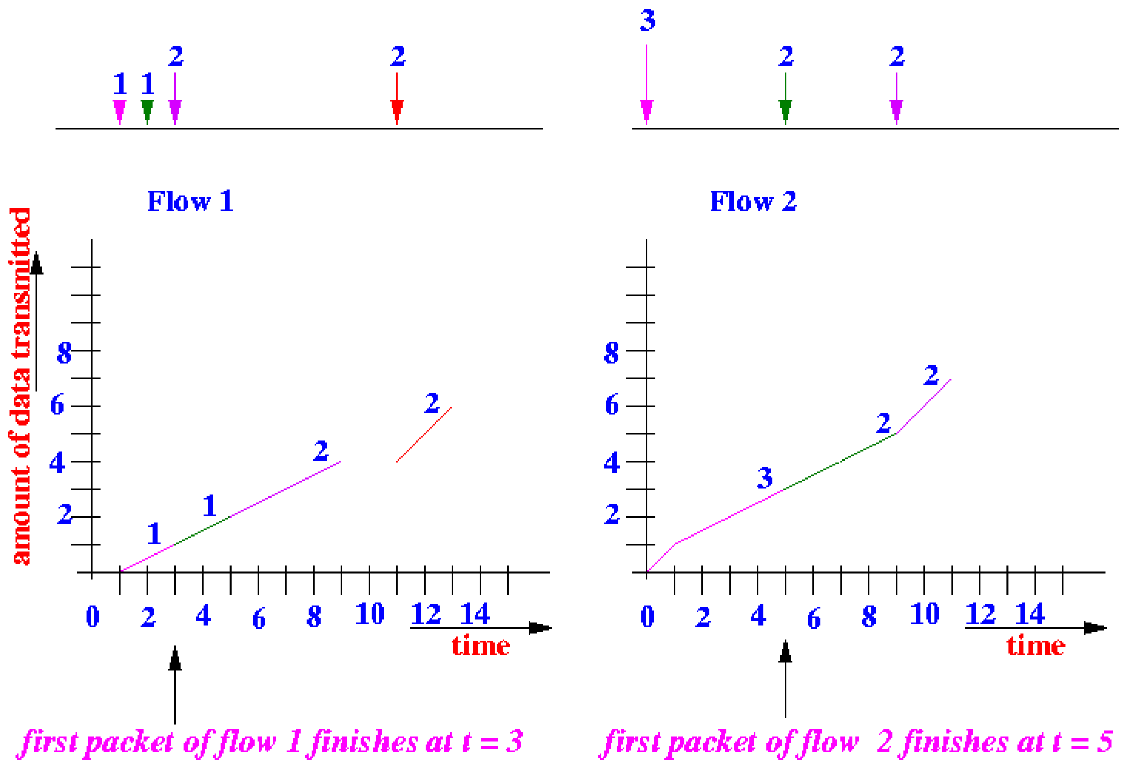


- The arrival and finish times of the packets are as follows:

	Flow 1:		FLow 2:	
	Arrival Time	Finish Time	Arrival Time	Finish Time
Packet 1:	1	4	0	4
Packet 2:	2	5	5	8
Packet 3:	3	9	9	11
Packet 4:	11	13	-	-

- **Packetized approximation for GPS:**
  - It is a general practice to assume that a packet has arrived when the **last bit** of the packet is received...

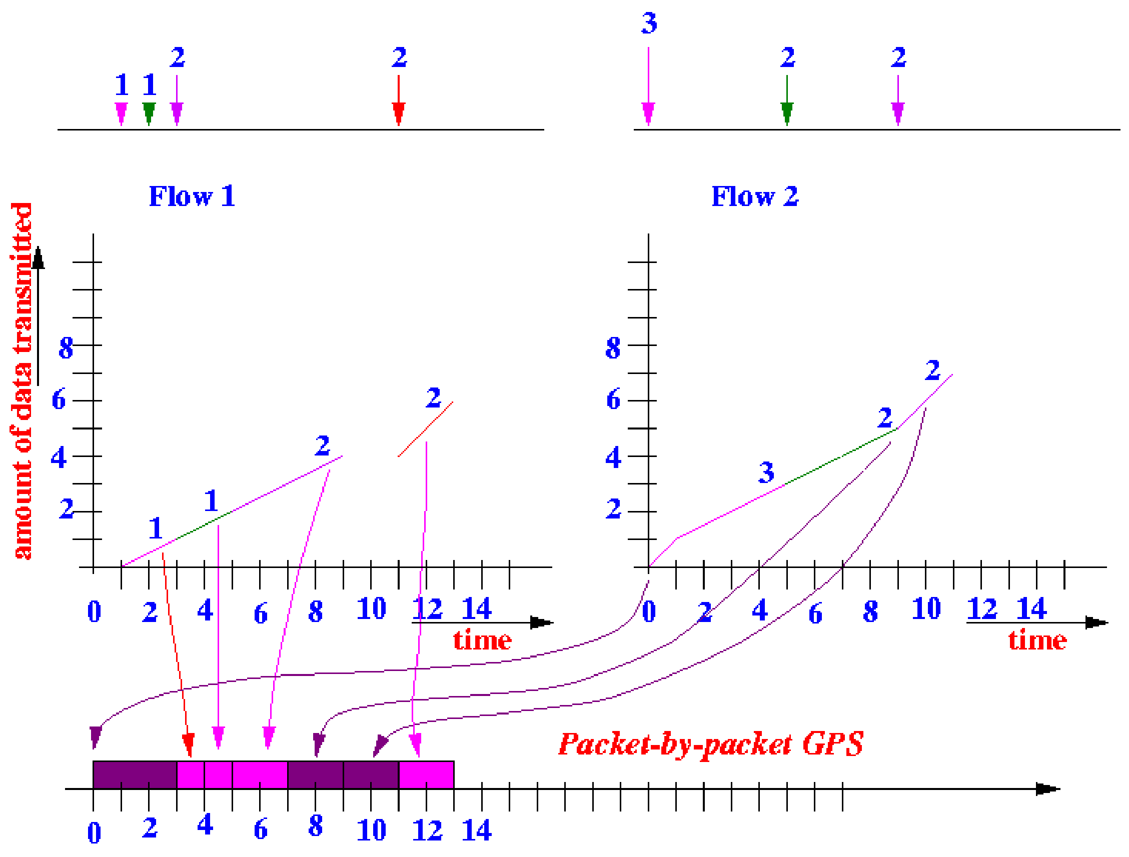
    Hence, packets arrive "instantaneously" at a router, just like the assumption in GPS...
  - Parekh proposed the following packetized approximation of the GPS server:
    - The Packet GPS (PGPS or WFQ) transmits each packet in its **entirety** before transmitting the next packet.
    - The packet selection criteria of the packet to be transmitted next by PGPS is the packet that has the **next smallest *finish time* in the GPS system**
  - This sounds like a plan, except you have to live with a shortcoming:
    - At the moment that the PGPS server picks the next packet for transmission (the one with the smallest finish time in the GPS system), that packet **may not have arrived yet !!!**
    - You can see it happening in the above examples:



    - According the the GPS system, the first packet of flow 1 will finish transmission first.
    - When the Packet-by-packet GPS server decides to transmits the first packet **at time 0**, the first packet of flow 1 **has not yet arrived** !
    - But packet 1 of flow 2 already has arrived at time 0.

- When the system is at time 0, it has to make a transmission decision **based on currently available information** and not based on future events.
- Thus, the PGPS server will pick packet 1 of flow 2 to start transmission at time 0 and **once the transmission of a packet is started, it cannot be interrupted** (atomic).

- **Examples of Packet-by-packet GPS:**
  - Example 1:



Packet transmission order:

1. Selection time = 0:

   packets present: (Flow 2, Packet 1, Fin Time = 5).
   Selected: (Flow 2, Packet 1)
   Note: eventhough (Flow 1, packet 1) has smaller finish time, this packet is not present at time 0, so (Flow 2, Packet 1) gets selected

   Time 0-3: transmits (Flow 2, Packet 1)

2. Selection time = 3:

   packets present: (Flow 1, packet 1, Fin Time = 3), (Flow 1, packet 2, Fin Time = 5)
   Selected: (Flow 1, Packet 1)

   Time 3-4: transmits (Flow 1, Packet 1)

3. Selection time = 4:

   packets present: (Flow 1, packet 2, Fin Time = 5)
   Selected: (Flow 1, Packet 2)

Time 4-5: transmits (Flow 1, Packet 2)

4. Selection time = 5:

   packets present: (Flow 1, packet 3, Fin Time = 9), (Flow 2, packet 2, Fin Time = 9)
   Selected: a tie, just pick one - (Flow 1, Packet 3)

   Time 5-7: transmits (Flow 1, Packet 3)

5. Selection time = 7:

   packets present: (Flow 2, packet 2, Fin Time = 9)
   Selected: (Flow 2, Packet 2)

   Time 7-9: transmits (Flow 2, Packet 2)

6. Selection time = 9:

   packets present: (Flow 2, packet 3, Fin Time = 11)
   Selected: (Flow 2, Packet 3)

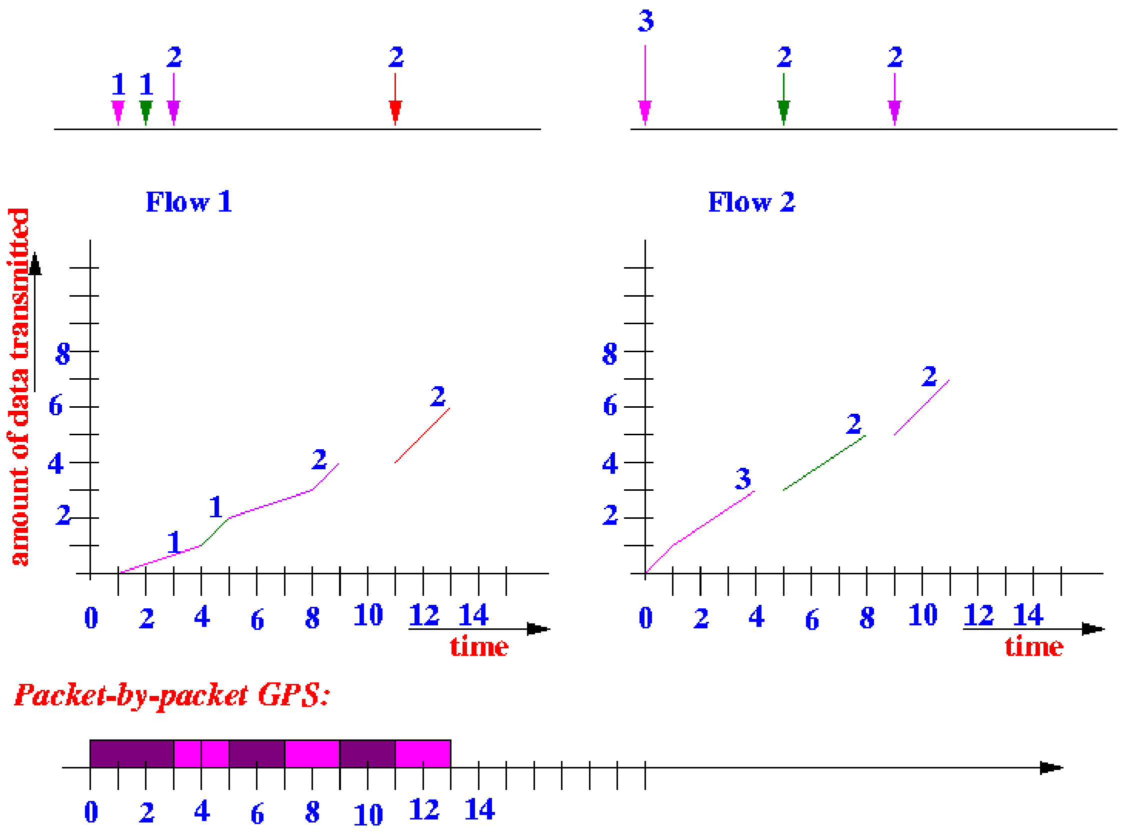
   Time 9-11: transmits (Flow 2, Packet 3)

7. Selection time = 11:

   packets present: (Flow 1, packet 4, Fin Time = 13)
   Selected: (Flow 1, Packet 4)

   Time 11-13: transmits (Flow 1, Packet 4)

- Example 2:



- The light-magenta rectangles represent the transmissions of the packets of flow 1

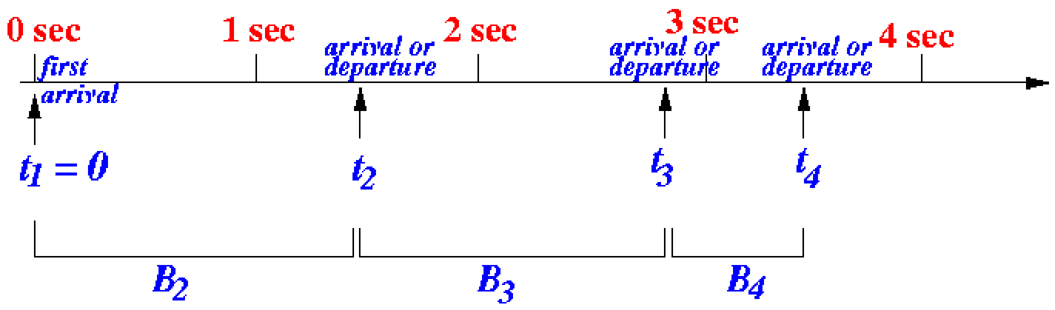
  - The dark-magenta rectangles represent the transmissions of the packets of flow 2
  - I won't give so much details in this example, you should be able to figure it out by yourself.

- I will skip the Theorem 1 and Theorem 2 - basically they say that although PGPS does not operate the same manner as GPS, the difference in behaviour between them is very small. Parekh proved that the difference the service received by any flow in PGPS is at most 1 packet less than the service that it would receive in GPS at any given time (Theorem 2).

  This is a very strong result, because 1 packet difference is also the best you can do (to get 0 packet difference you will need to transmit packets non-atomically as in GPS - so 1 is the best you can do)

- **A Virtual Time implementation of WFQ - Events & event times**

  - Clearly, we must find a better way to calculate the finish time of packets than drawing pictures...
  - Parekh has devised the following ingenious way to implement the Packet-by-packet GPS scheduling.
  - Recalled that we **normalized** the transmission speed by the output rate ***R***:
    - Normalize the packet lengths so that the transmission (service) rate is equal to ONE (1) - see page 348, 7th line from below)
  - An **event** is a packet arrival or a packet departure
    - Notice that a packet has arrived when the **last bit** is received
    - A packet has departed when the **last bit** is sent
    - Hence, an event is always **instantaneous**
    - Parekh found a way to implement the PGPS system by making computational adjustments **only** when an event occurs ! (instead of having to draw graphs like before)
    - Denote the times when the j^{th} event occurs by t_j.



  - Notice that there are no events (no arrivals and departures) between two consecutive event, the set of **active** flows will remain constant between 2 consecutive event times.
  - B_j = the set of active flows during the (real time) interval (t_{j-1}, t_j).

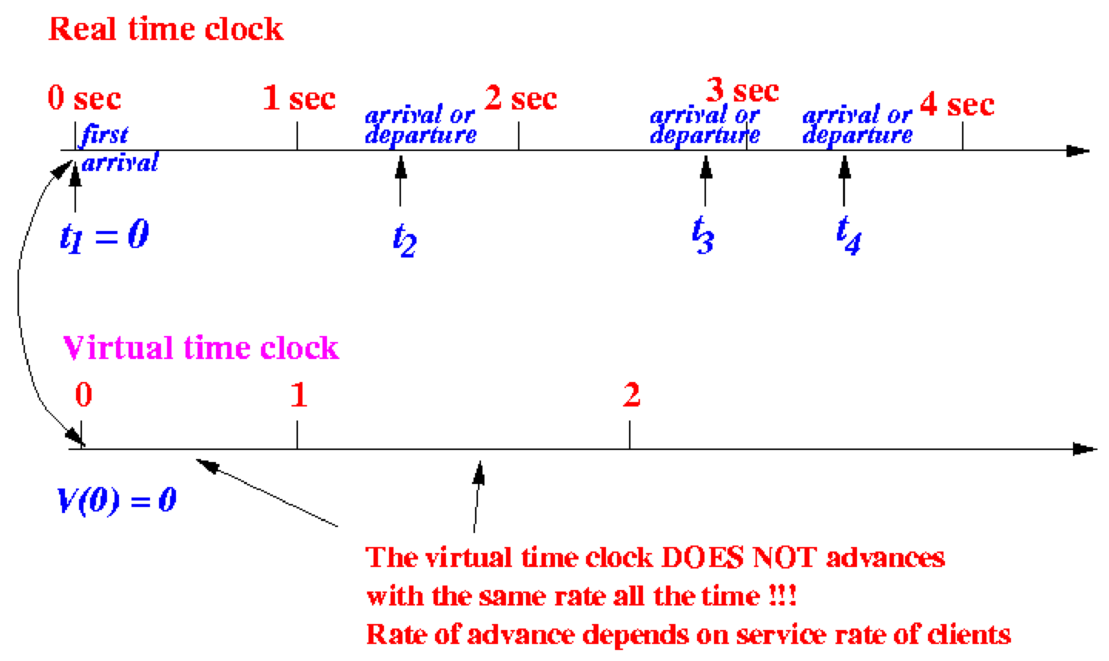
    Note: a flow is **active** is it has **some packet waiting** in the router to be transmitted.
  - I will call the interval B_j the *j^{th} event interval*

- **A Virtual Time implementation of WFQ - Virtual Time**

  - Parekh defines a **virtual time** system that has a **very ingeneous** "time keeping mechanism" that will make implementing Packet-by-packet GPS very simple.

    In the "normal" earth time, the **rate of increase in time value** (or the speed of the clock) is **contant** - (look at your watch, the second hand jumps ahead very regularly).

Parekh defines a clock that has **rate of increase** (or the speed of the clock) that is different at different situations - if you are a fan of relativity theory or had the opportunity to travel near the speed of light, you will delight in this discussion....



- There is a one-to-one correspondence between the real time clock value and the virtual clock time value, but the correspondence is rather complicated...
- The real time is denoted by ***t*** and the virtual time that corresponds to the real time ***t*** is denoted by ***VT(t)***
- How to start time keeping:
  - The (real) time clock start running when the first packet arrives, i.e.: $t_1 = 0$

    (It makes sence, before the first packet arrival, the system is idle and we do not to do any time keeping exercise.)
  - At this time, the virtual time clock also starts running: ***VT(0) = 0***

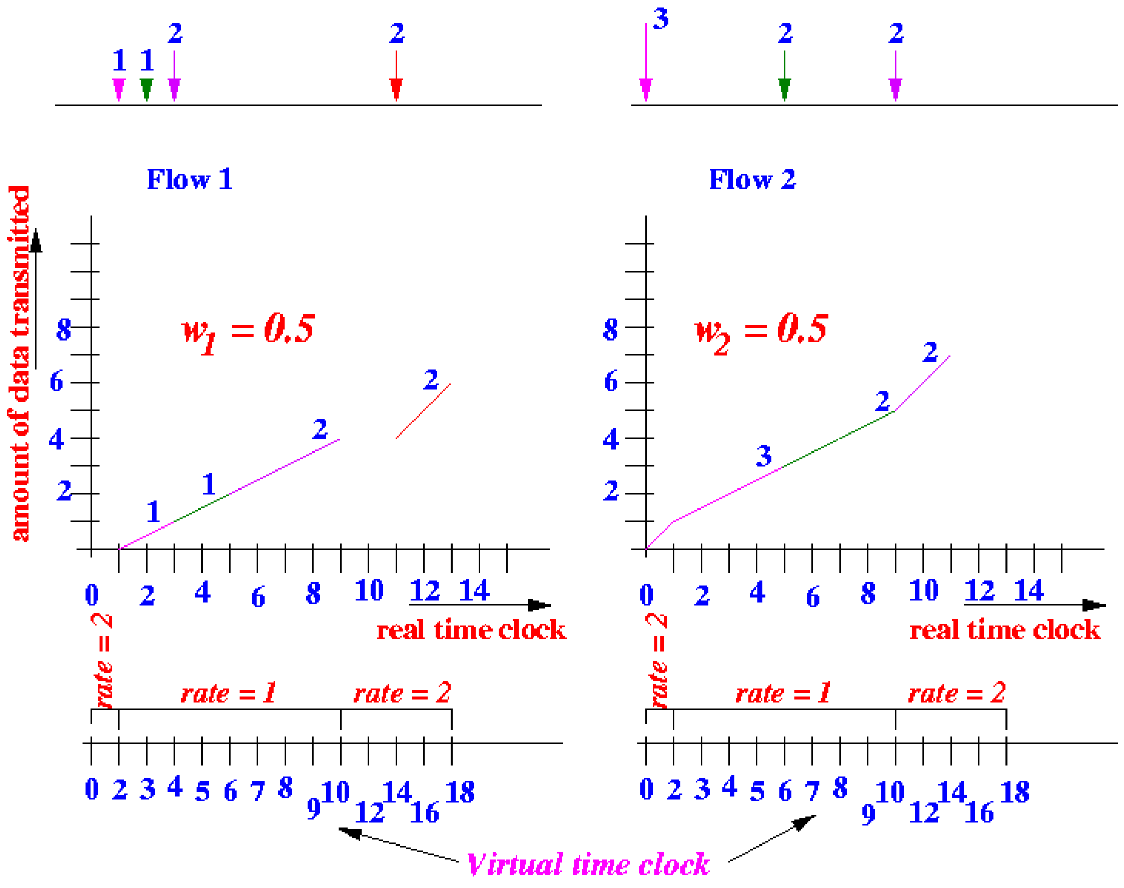
  So the real time clock and the virtual time clock start ticking (begins to run) when the first packet arrives.
- **How does the virtual time clock advance ?**
  - Recall that w_i is the weight assigned to flow ***i*** and **w_i < 1**
  - The virtual time changes as follows:
    - The virtual time changes with a constant rate **within each event interval B_j**
    - So the virtual time can advance with **different rates** in **different event intervals**
    - The virtual time advances in the *j^{th} event interval B_j* as follows:

```
                                     x
VT(t_{j-1} + x) = VT(t_{j-1}) + ----------------
                                 sum_{i in B_j} w_i
```

(Recall that B_j is the set of active flows in the event interval (t_{j-1}, t_j).

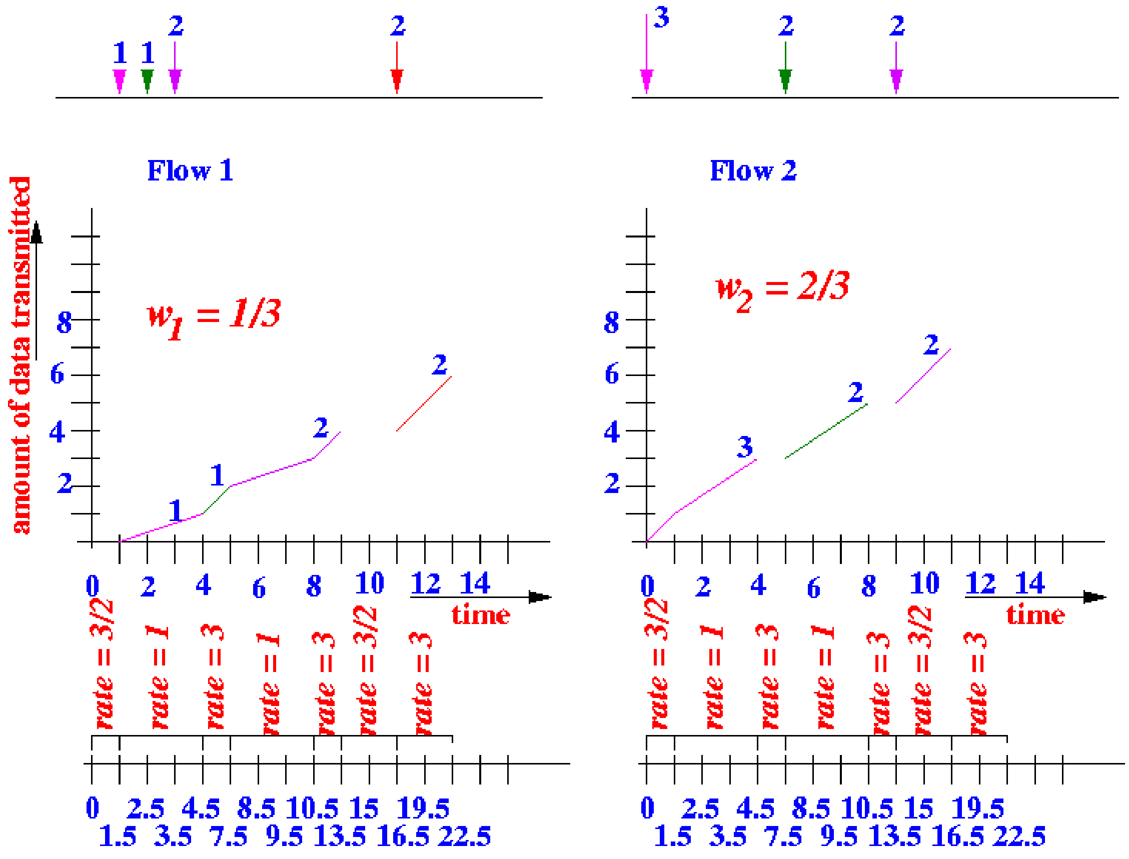
- **Example Virtual Time keeping 1:**

  The following figure shows the advancement of the virtual time clock in the packet transmission example 1:



- **Example Virtual Time keeping 2:**

  The following figure shows the advancement of the virtual time clock in the packet transmission example 2:

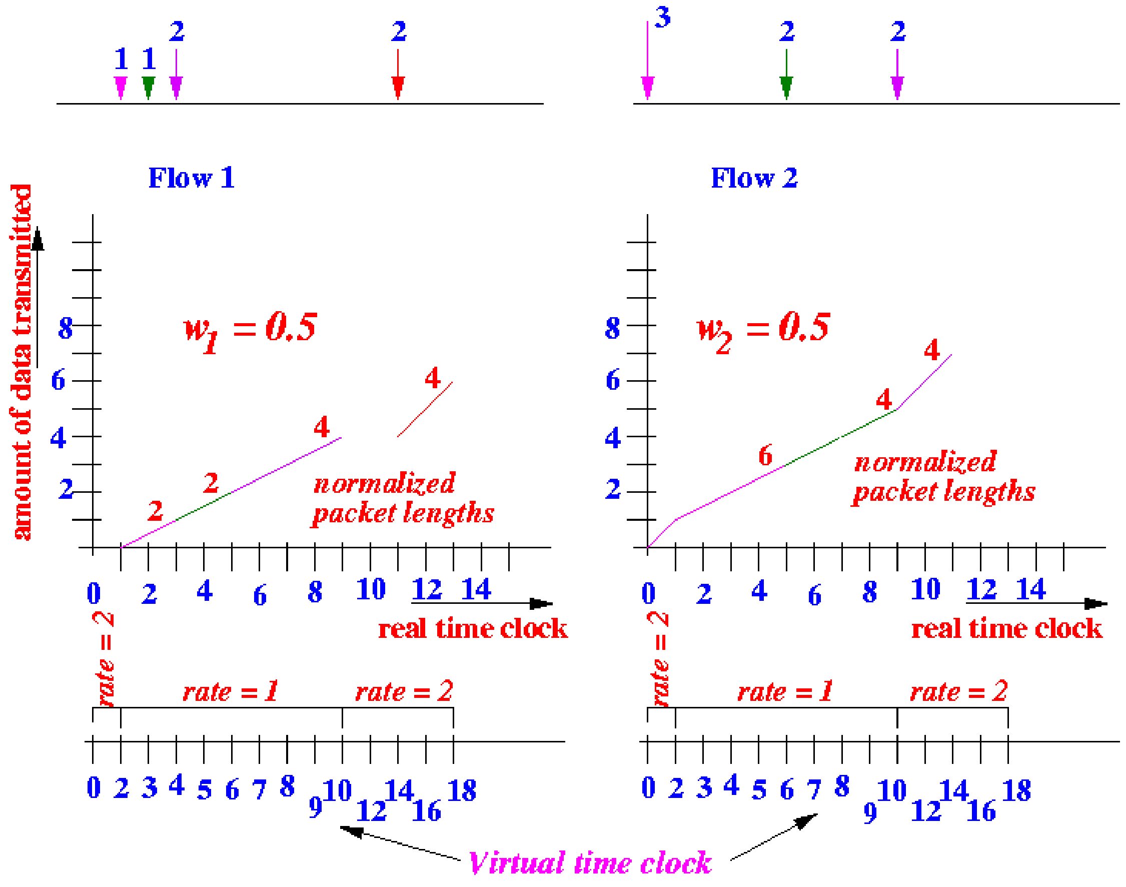


- **$6,000,000 question:**
    - Why on earth would you want to make the clock so difficult to track ???
    - Answer: it is tracking the **progress of the packet transmissions**
- **Normalized packet length**
    - Let us define the "normalized work load" as:

```
                                    Length of packet of flow i
Normalize packet length of flow i = -------------------------
                                                wi
```

    - This is quite intuitive: since flow ***i*** received the fraction ***w_i*** of service, it will take ***$1/w_i$*** times longer to process packets of flow ***i***.
- **Example Virtual Time keeping with normalized packet length 1:**

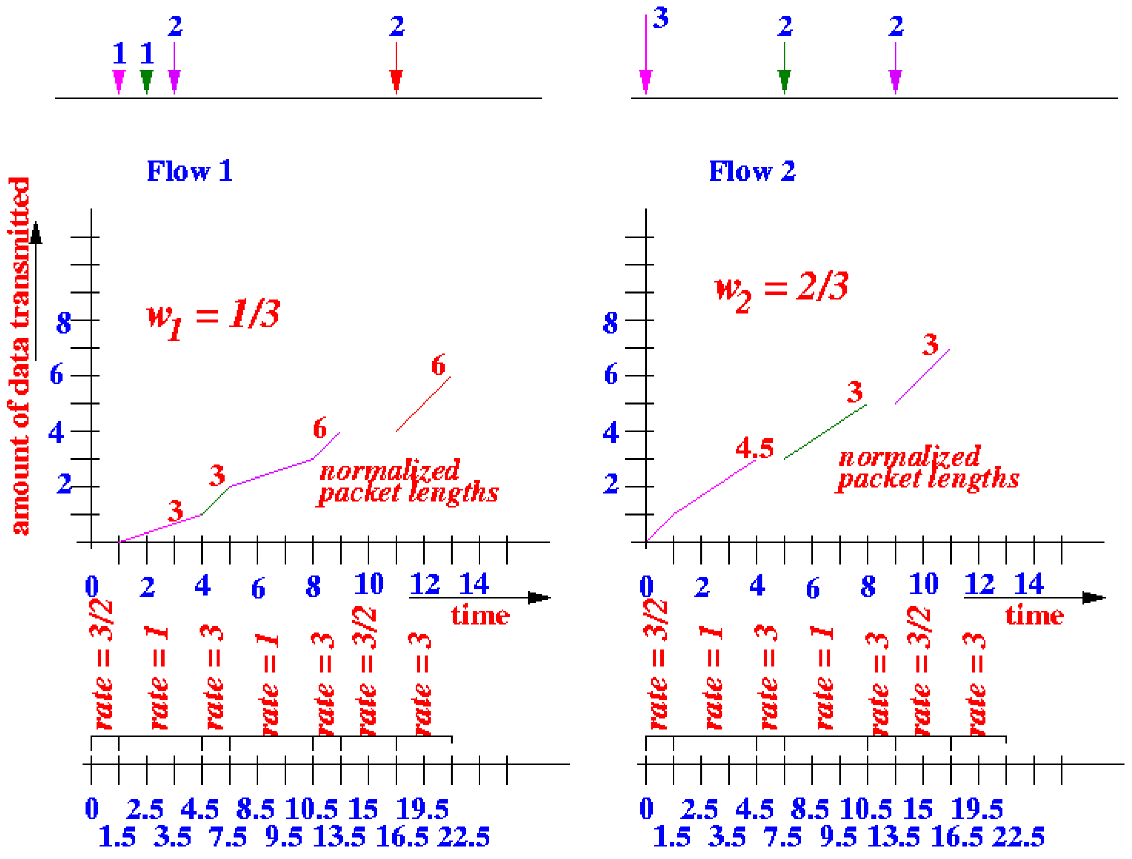
The following figure shows the advancement of the virtual time clock along with the normalized packet length in example 1:



Notice that the difference of the ending virtual time and the starting virtual time of every packet is equal to the normalized length of that packet !

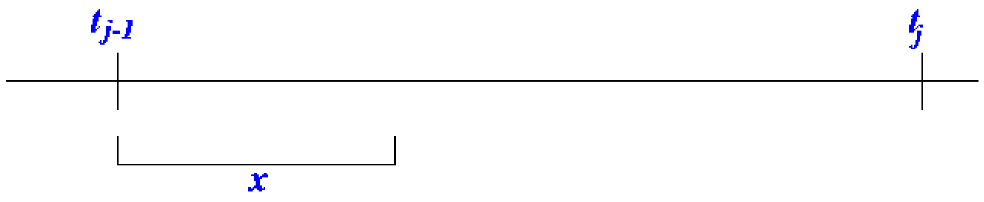
- **Example Virtual Time keeping 2:**

  The following figure shows the advancement of the virtual time clock in the packet transmission example 2:



Notice again that the difference of the ending virtual time and the starting virtual time of every packet is equal to the normalized length of that packet !

- **Why does it work ?**
  - Consider the progress of a flow ***k*** during **any** event interval ***[t_{j-1}, [t_j]***:



  - As given above, the rate of increase in the **virtual time** is as follows:

$$VT(t_{j-1} + x) = VT(t_{j-1}) + \frac{x}{\text{sum\_\{i in } B_j\text{\} } w_i}$$

    Thus, for every ***x*** units of **real time**, the **virtual time** increases by ***x/sum_{i in B_j} w_i***

  - Consider now the rate of service received by flow ***k*** during these ***x*** units of **real time**.

    The data transmission rate received by flow ***k*** is equal to:

$$\frac{w_k}{\text{sum\_\{i in } B_j\text{\} } w_i}$$

So in *x* units of **real time**, the router has has transmitted this amount of data from flow ***k***:

$$\frac{w_k}{\text{sum\_}\{i \text{ in } B_j\}\ w_i}\ x$$

- Let us now compute the data trasnmission rate per **virtual time unit**:

  In ***x/sum_{i in B_j} w_i*** units of **virtual time**, the router has transmitted

  $$\frac{w_k}{\text{sum\_}\{i \text{ in } B_j\}\ w_i}\ x$$

  amount of data, so the "virtual transmission data rate" is equal to: w_k units per virtual time unit.

  Notice this is **constant** no matter how many flows are active !!!

- Theirefore, if a packet of flow ***k*** has ***L*** bits, the amount of **virtual time** that is used to transmit this packet is equal to:

  $$\frac{L}{w_k}$$

  And this is the behavior you see in the plots above...

- **A Virtual Time implementation of WFQ - Putting it together**

  - Parekh devised an implementation of the Packet-by-packet GPS scheduling method by using the above principle where a packet of length ***L*** of flow ***j*** is always transmitted in L/w_j units of **virtual time**

  - You need one more piece of information to understand his method: when does a packet of flow ***j*** begins its service ?

    - If flow ***j*** was **idle** (not active) when a new packet of flow ***j*** arrives, then the new packet will begin service **immediately**

    - If flow ***j*** was **active** when a new packet of flow ***j*** arrives, then the new packet will begin service **after the last packet of flow *j*** finishes.

  - Believe it or not, you can determine the exact **virtual finish time** in **both cases**:

    - If flow ***j*** was **idle**, then the virtual time is the **current** value of the virtual clock (that can be determined because we keep track of the current real time and the current virtual time)

    - If flow ***j*** was **active** when a new packet of flow ***j*** arrives, then we use the virtual finish time of the last packet of flow ***j*** to compute the virtual finish time of the new packet....

  - The method designed by Parekh is as follows:

    - Let p_j^i denote the ***i***th packet of flow ***j***.

    - Each packet p_j^i is assigned a **virtual finish time** $V(p_j^i)$ which is also the value on which the packets are ordered for transmission.

    - We define: $V(p_j^0) = 0$

    - When a new packet arrives at (real time) ***t***, the **virtual finish time** of the new packet is computed using the formula:

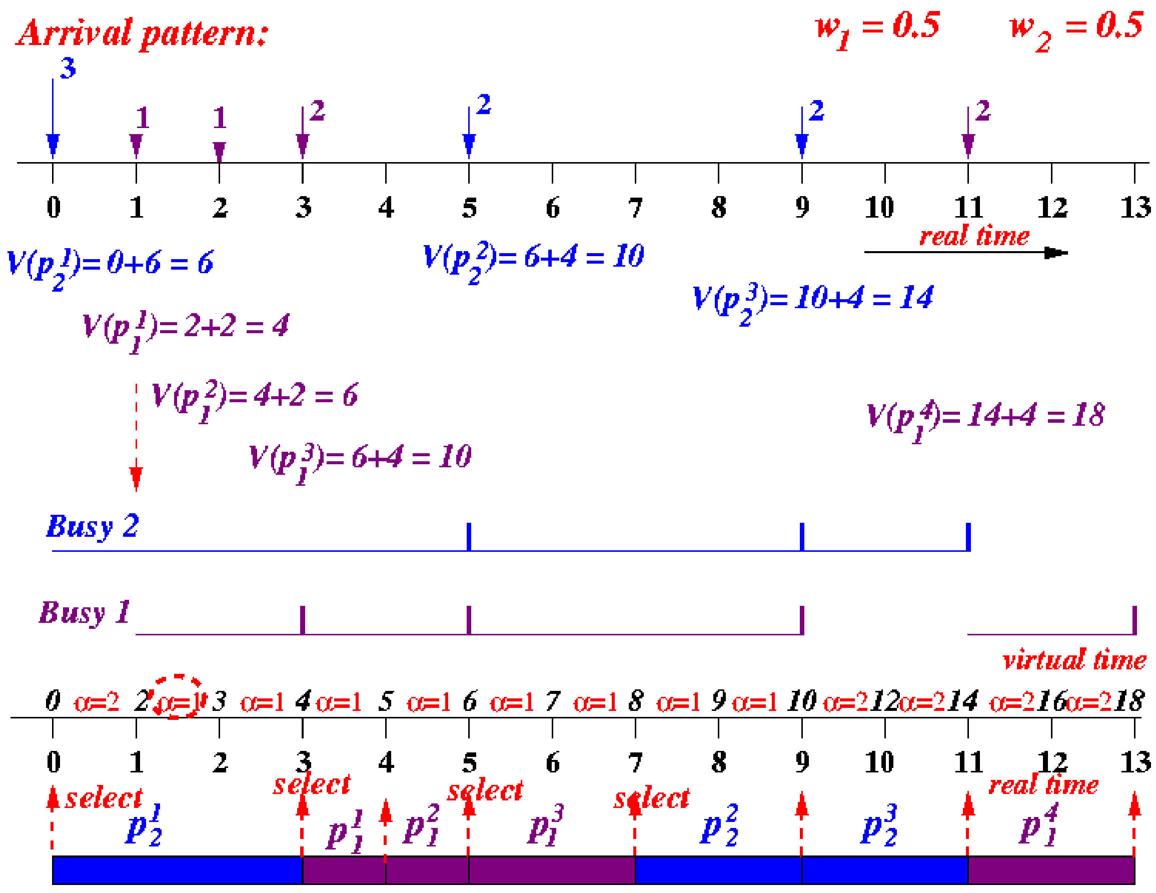
L_j^i

```
V(pj^i) = max(VT(t), V(pj^i - 1)) + -----
                                    wj
```

where ***VT(t)*** is the value of the virtual time clock at the (arrival time) ***t***.

- The packets are transmitted by increasing virtual finish time values.

- **Example 1**
  - The following figure shows how the virtual finish time are computed for the arrival pattern used for the case where $w_1 = 0.5$ and $w_2 = 0.5$:



- Computing the virtual finish times are pretty straightforward using the formula...
- The most important thing you have to know in PGPS is the fact that the **active flows** is determined by the **virtual finish times** of last packet of each of the flows.

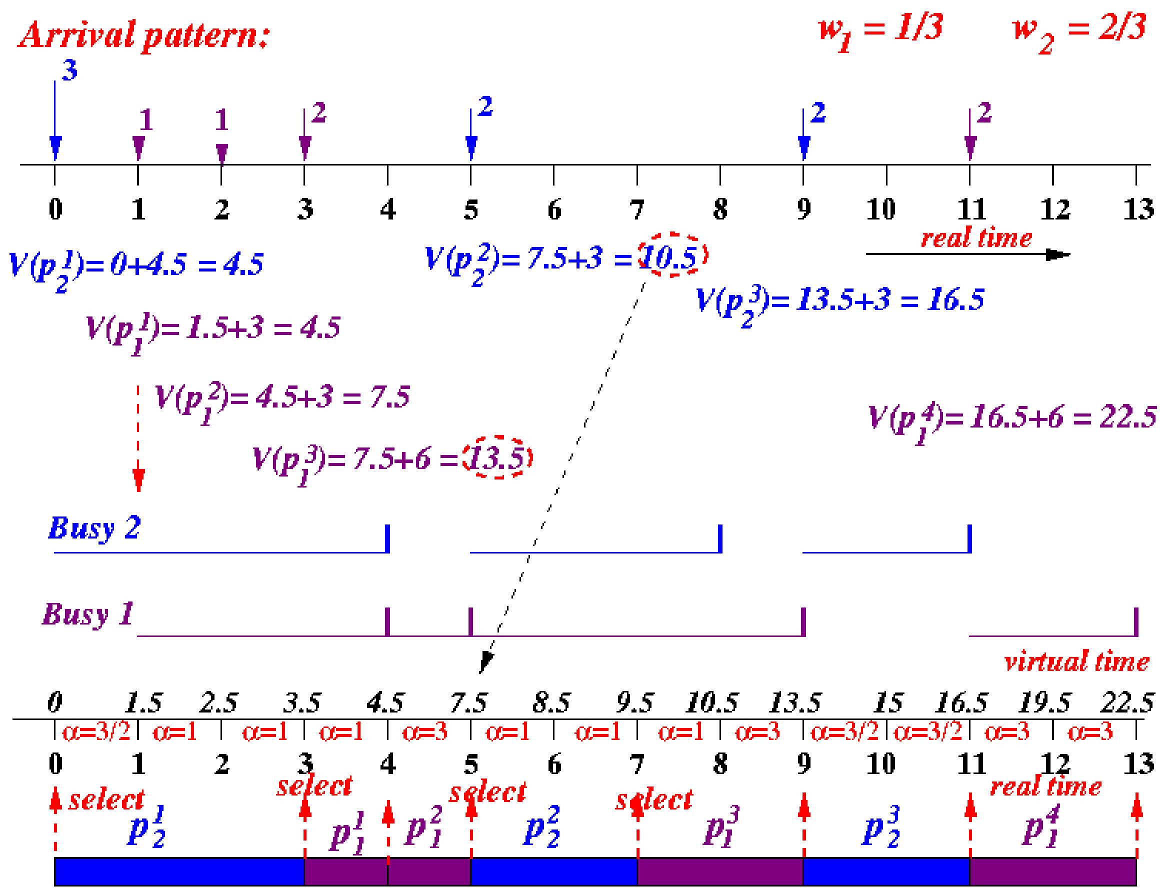
  So, a flow may be **active** at a certain time while there are no packets in the system !

  For example, during the interval [3,5], flow 2 is **active**. But packet 1 of flow 2 has already departed from the system at time 3 !

  But since packet 1 of flow 2 has a **virtual finish time** of 6, flow 2 "remains" active until **virtual time** 6 (which is real time 5).

- **Example 2**
  - The following figure shows how the virtual finish time are computed for the arrival pattern used for the case where w_1 =

1/3 and $w_2 = 2/3$:


Arrival pattern:
$w_1 = 1/3$
$w_2 = 2/3$
3
1
1
2
2
2
2
0 1 2 3 4 5 6 7 8 9 10 11 12 13
real time
$V(p_2^1) = 0+4.5 = 4.5$
$V(p_2^2) = 7.5+3 = 10.5$
$V(p_2^3) = 13.5+3 = 16.5$
$V(p_1^1) = 1.5+3 = 4.5$
$V(p_1^2) = 4.5+3 = 7.5$
$V(p_1^4) = 16.5+6 = 22.5$
$V(p_1^3) = 7.5+6 = 13.5$
Busy 2
Busy 1
virtual time
0 1.5 2.5 3.5 4.5 7.5 8.5 9.5 10.5 13.5 15 16.5 19.5 22.5
α=3/2 α=1 α=1 α=1 α=3 α=1 α=1 α=1 α=3 α=3/2 α=3/2 α=3 α=3
0 1 2 3 4 5 6 7 8 9 10 11 12 13
select
select
select
select
real time
p_2^1 p_1^1 p_1^2 p_2^2 p_1^3 p_2^3 p_1^4