Link Scheduling

Dr. Yeali S. Sun
National Taiwan University

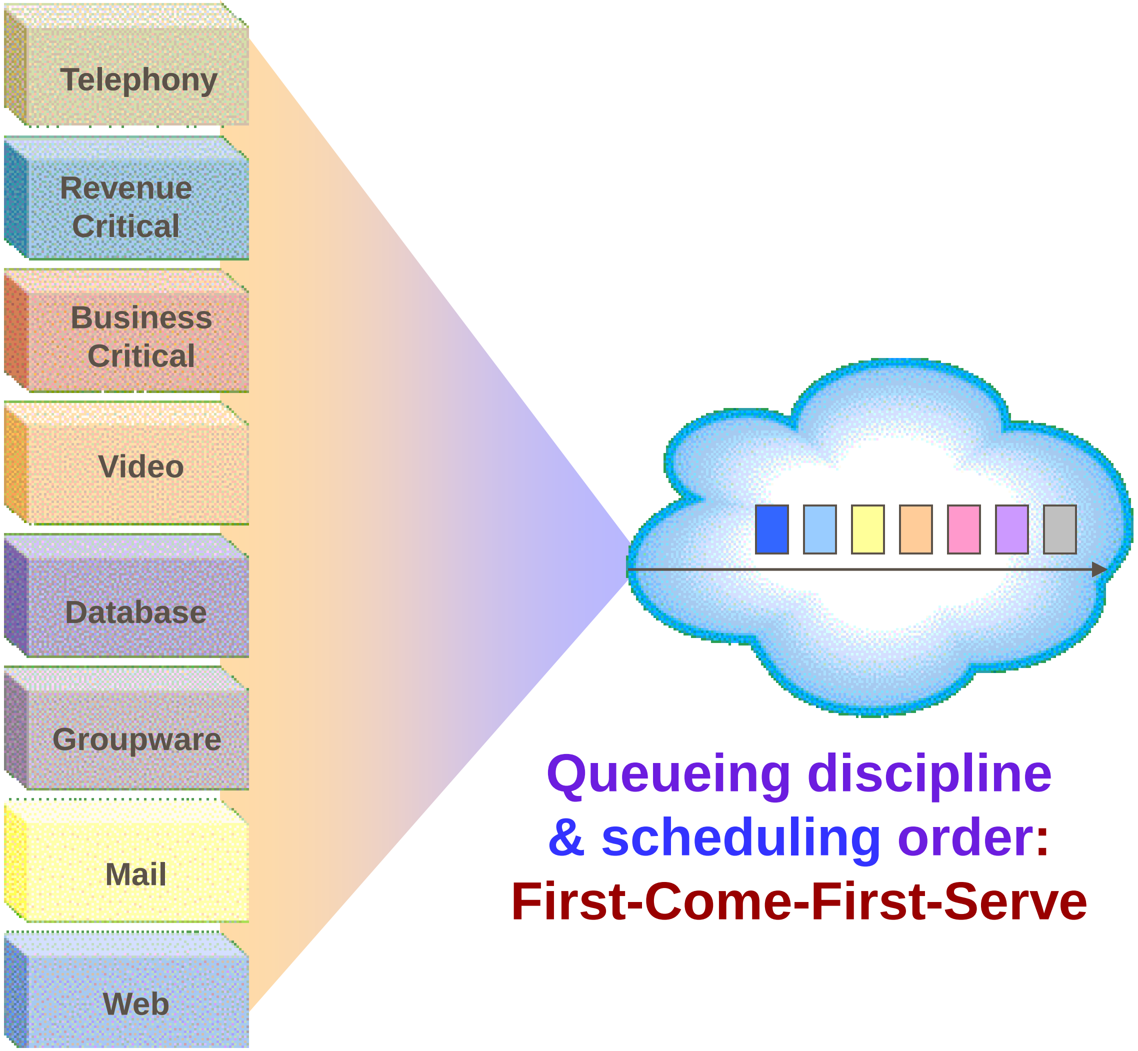
Link Scheduling in Integrated Services Networks (1/4)

- Traditionally, the ***flexibility*** of data networks has been *traded off* with the ***performance guarantees*** given to its users, e.g.,
  - The **telephone network** provides *good* performance guarantees but *poor flexibility*
  - **Packet switched networks** are more flexible but only provide marginal performance guarantees.
  - Traffic characteristics
- Integrated services networks will carry *a wide range of traffic types* and must be able to provide performance guarantees to real-time sessions such as voice and video.

-> The problem is how to reconcile these apparently conflicting demands when the **short-term demand** for link usage frequently exceeds the usable **capacity**.

Today's Internet

- Performance of *mission critical* applications are threatened
- *Uncontrolled* use of bandwidth in WAN
- Need technologies to manage link sharing and guarantee QoS on a per interface basis
- Need *automated* QoS management



Queueing discipline & scheduling order: First-Come-First-Serve

Link Scheduling in Integrated Services Networks (2/4)

- Schedule packet transmissions of the **sessions** (flows) at a single node.
- Packet delay in the network can be expressed as the sum of the processing, queueing, transmission, and propagation delays
- The **focus** is on how to **limit *queueing delay***.
- Wish to guarantee ***worst-case packet delay***.

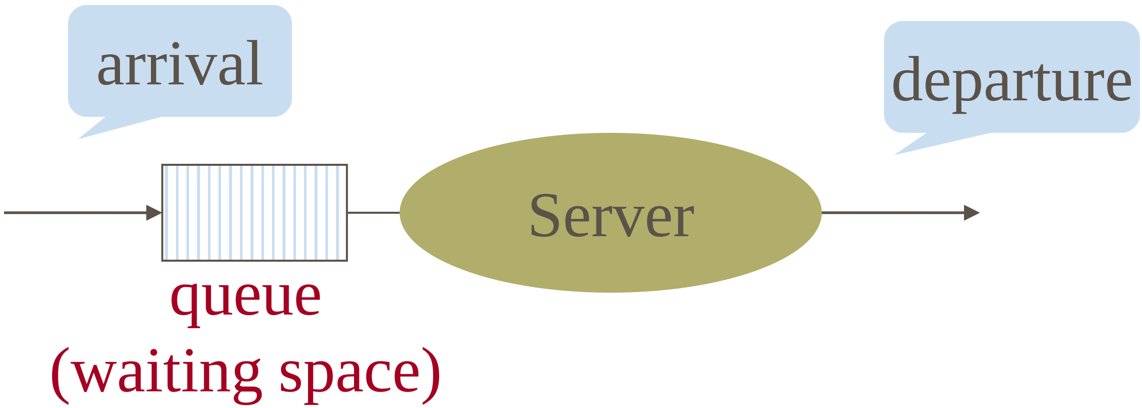
Link Scheduling in Integrated Services Networks: requirements (3/4)

- Treat users **differently**, in accordance with their desired **QoS.**
- Flexibility should not compromise the **fairness** of the scheme (e.g., in priority-based scheduling).
- Performance guarantees should be analyzable.

Link Scheduling in Integrated Services Networks (4/4)

- An important approach is to *combine* the use of a *packet service discipline* based on ***Generalized Processor Sharing (GPS)*** *and Leaky Bucket rate control* to provide
  - ***flexible***, ***efficient***, and ***fair*** use of the links, and
  - ***performance guarantees***
- **Weighted Fair Queueing (WFQ)** is the **packet** version of GPS which closely *approximates* GPS.
  - a way of rate-based flow control

A Link is modeled as a Queueing Server

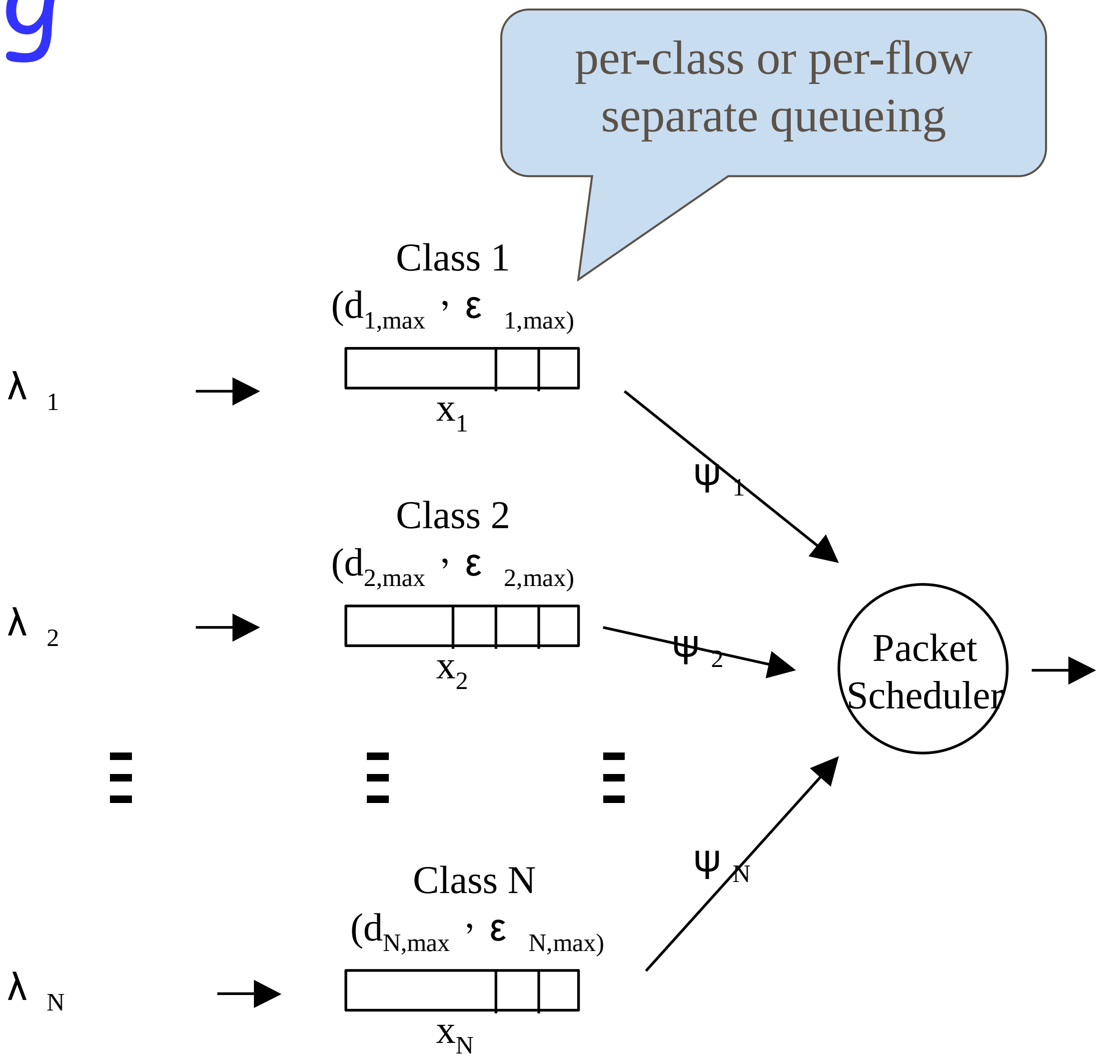


- **Arrival process**
  - customers to be served
  - **Inter-arrival time** distribution
- **Queue**
  - **finite** or infinite capacity
- **Service time distribution**
  - **Workload**
- **Queueing discipline**
  - FIFO, LIFO, priority, etc.
- **Number of servers**

-> Nobody likes to wait in line.

Fair Queueing

- Consider **N** queues
- The goal is to provide *flexible*, *efficient* and *fair* use of the links
  - Flexible: meeting QoS of *all* queues
  - Efficient: *maximal* link utilization (work conserving)
  - Fair: *excess* bandwidth sharing and assignment



Generalized Processor Sharing (GPS)

- Head-of-line Processor Sharing service (PS)
  - A **separate** FIFO queue for **each** *session* sharing the same link
  - During any time interval, if there are exactly N packets at the head of the queues, each receives a **1/N** of the link speed
- GPS is a **generalized** form of PS

GPS Server (1/7)

- Consider a **work-conserving** server with rate *r* serving *N* sessions
  - *Work-conserving* means the server (the link) will **not** be idle if a packet **waiting** for transmission
- Each **session** *i* is assigned a fixed real-valued positive parameter ϕ_i.

GPS Server (2/7)

- $\phi_1, \phi_2, \ldots \phi_N$ are ***relative*** *amount of service to each session* such that let $S_i(\tau, t)$ be the amount of session *i* traffic *served* during an interval [τ, t] then

$$\frac{S_i(\tau, t)}{S_j(\tau, t)} \geq \frac{\phi_i}{\phi_j}$$

  - assuming any session *i* that is continuously backlogged in the interval [τ, t]

GPS Server (3/7)

- Sum up all sessions j

$$S_i(\tau,t)\sum_j \phi_j \geq (t-\tau) r \phi_i$$

$$S_i(\tau,t)\phi_j \geq S_j(\tau,t)\phi_i$$

$$S_i(\tau,t)\sum_j \phi_j \geq \phi_i \sum_j S_j$$

$$S_i(\tau,t)\sum_j \phi_j \geq \phi_i (t-\tau) r$$

$$g_i \geq \frac{\phi_i}{\sum_j \phi_j} r$$

GPS Server (4/7)

- Let $B_{GPS}(\tau)$ be the set of **backlogged** sessions at time τ
- If $B_{GPS}(\tau)$ remains **unchanged** during the interval, the service rate of session *i* during the interval will be exactly

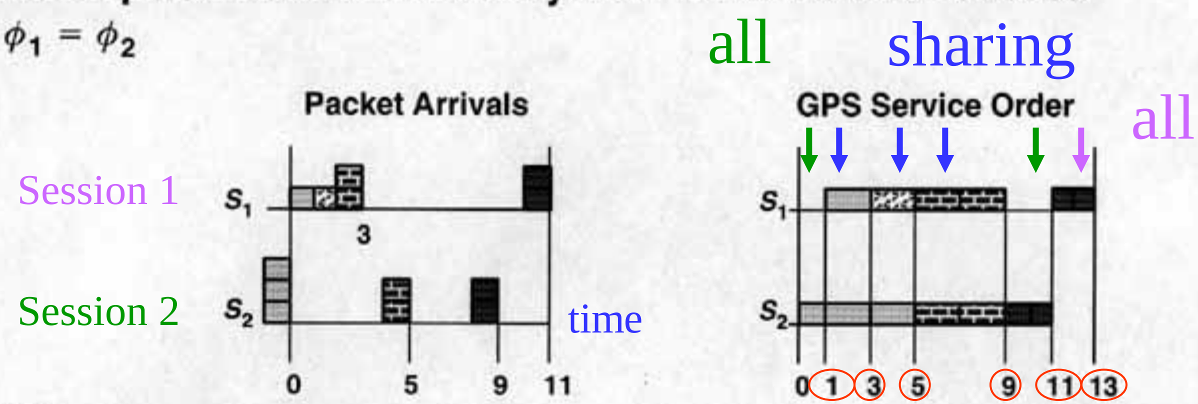
$$g_i = \frac{\phi_i}{\sum_{j \in B_{GPS}(\tau)} \phi_j} r$$

  - where *r* is the rate of the link

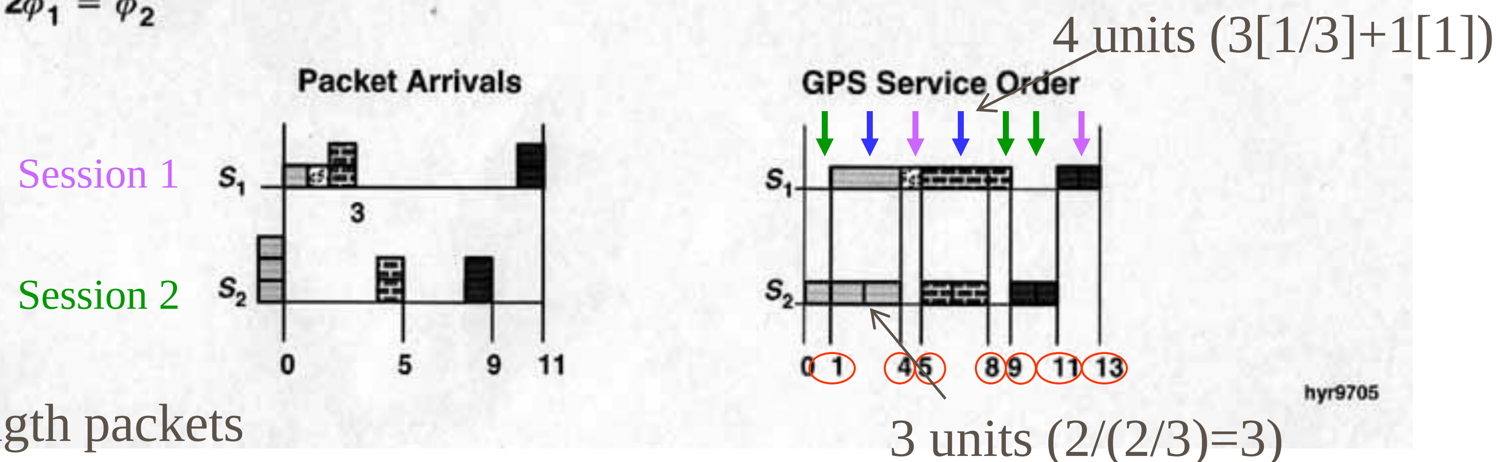
GPS Server: Fluid-flow (bit-by-bit) scheme (5/7)

Concurrent (shared) or Full possession!

- Consider a server with rate 1 serving 2 sessions
- Assume a packet has arrived only after its last bit has arrived
  - $\phi_1 = \phi_2$



  - $2\phi_1 = \phi_2$



Variable length packets

GPS Server: properties (6/7)

- Throughput guarantee
  - Define r_i be the session *i* average rate
  - As long as $\mathbf{r_i <= g_i}$, the session is guaranteed a throughput of ρ_i, *independent of the demands of the other sessions.*
- Delay bound
  - The delay of an arriving session *i* is bounded as a function of the session *i* **queue length**, **independent** of the queues and arrivals of the **other** sessions.
  - Schemes such as FCFS, LCFS, and strict priority do not have this property.

GPS Server: properties (7/7)

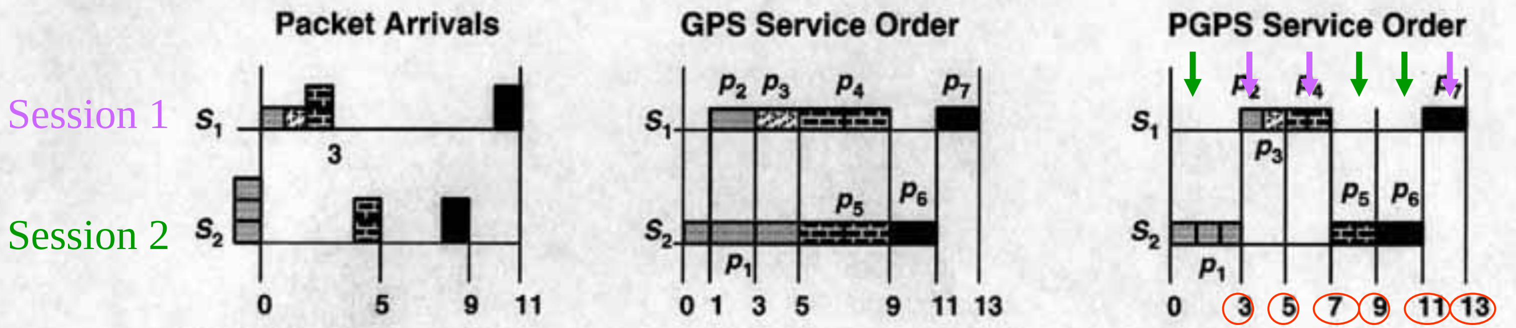
- By **varying the ϕ_i's**, the scheme has the flexibility of treating the sessions in a variety of ways, e.g.,
  - When all ϕ_i's are equal -> uniform processor sharing
  - When combined average rate of the sessions is less than r, any assignment of positive yields a **stable** system.
  - A high-bandwidth-delay-insensitive session *i* can be assigned g_i much less than its average rate, thus allowing for better treatment of the other sessions.
- **Worst-case delay guarantee**
  - When sources are constrained by leaky buckets.
  - **Attractive** for sessions with **real-time** constraints like voice and video

Packet Generalized Processor Sharing (PGPS)

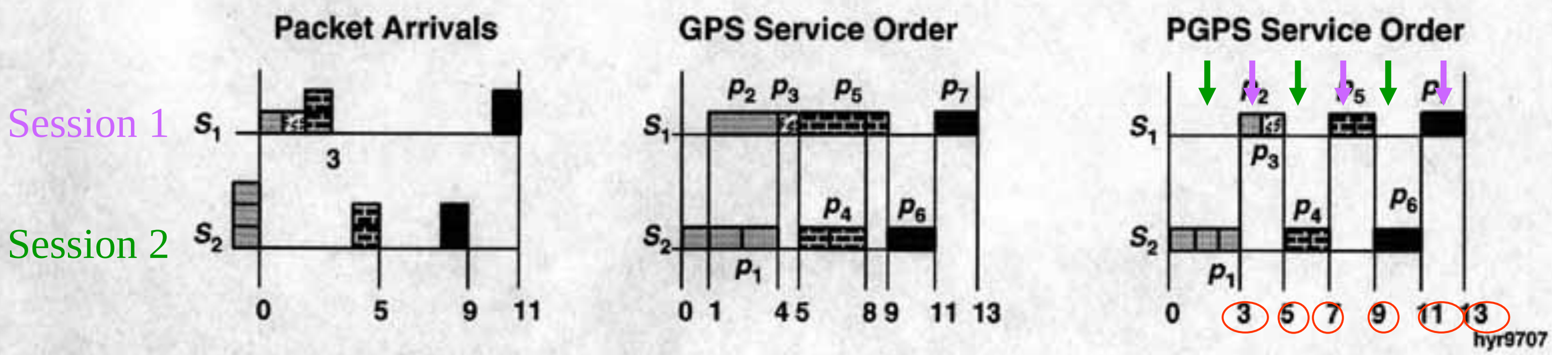
- GPS
  - **Fluid-flow (bit-by-bit)** scheme (concurrency)
  - Cannot be applied to packet-based networks
- PGPS
  - A **packet** approximation algorithm of GPS
  - Packet-by-packet scheme
  - a.k.a Weighted Fair Queueing (WFQ)

PGPS Server: packet version of GPS

- Consider a server with rate 1 serving 2 sessions
- Assume a packet has arrived only after its last bit has arrived
  - $\phi_1 = \phi_2$



  - $2\phi_1 = \phi_2$



Packet Generalized Processor Sharing (PGPS): Properties

1. ***Weighted*** fair allocation of bandwidth
2. ***Minimum*** bandwidth guarantee
3. ***Flow isolation***
   - Protection from misbehaving sources such as UDP flows that do not reduce rate when congestion occurs
4. ***Guaranteed bounded delay* services**
   - Provided sources are *leaky bucket constrained.*

WFQ: Packet Scheduling

- WFQ tries to ***emulate*** GPS.
- Consider ***two*** queueing systems
  - one using the GPS discipline and
  - one using the PGPS discipline
- Determine which packet to server **next**?
  - Serve packets in increasing order of d_p^{GPS}
    - d_p^{GPS} : the departure time of packet p under GPS

WFQ: Packet Scheduling (cont'd)

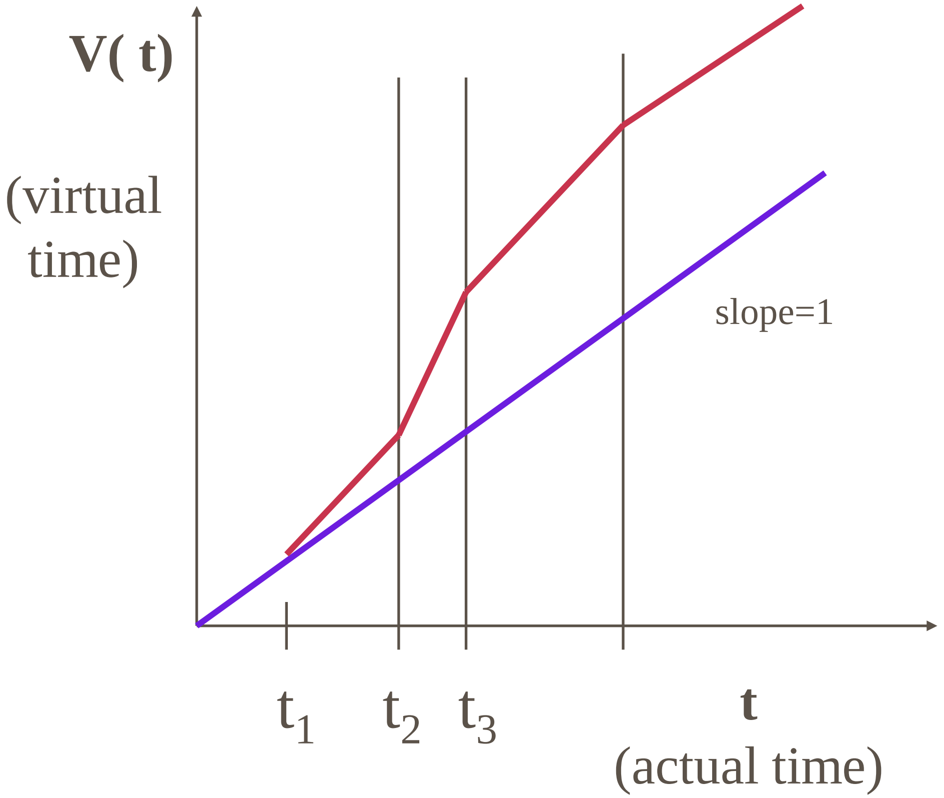
- Under GPS, when the system is ready to choose the next packet to transmit, the **next packet to depart under GPS** may **not** have arrived at the packet system yet.
  - Fluid model vs. packet model
- Waiting for it may cause system **idle** under non-empty system, i.e. non-work conserving.

PGPS: Virtual Time - notations

- Assume server works at rate 1
- ***Event***
  - a ***packet arrival*** and ***departure*** from the ***GPS server***
- t_j : the time at which the j^{th} event occurs
  - assume $t_1=0$
- **B_j**: the set of sessions that are backlogged in the interval (t_{j-1}, t_j)
- **V(t)**: *zero* for all times when the server is *idle*

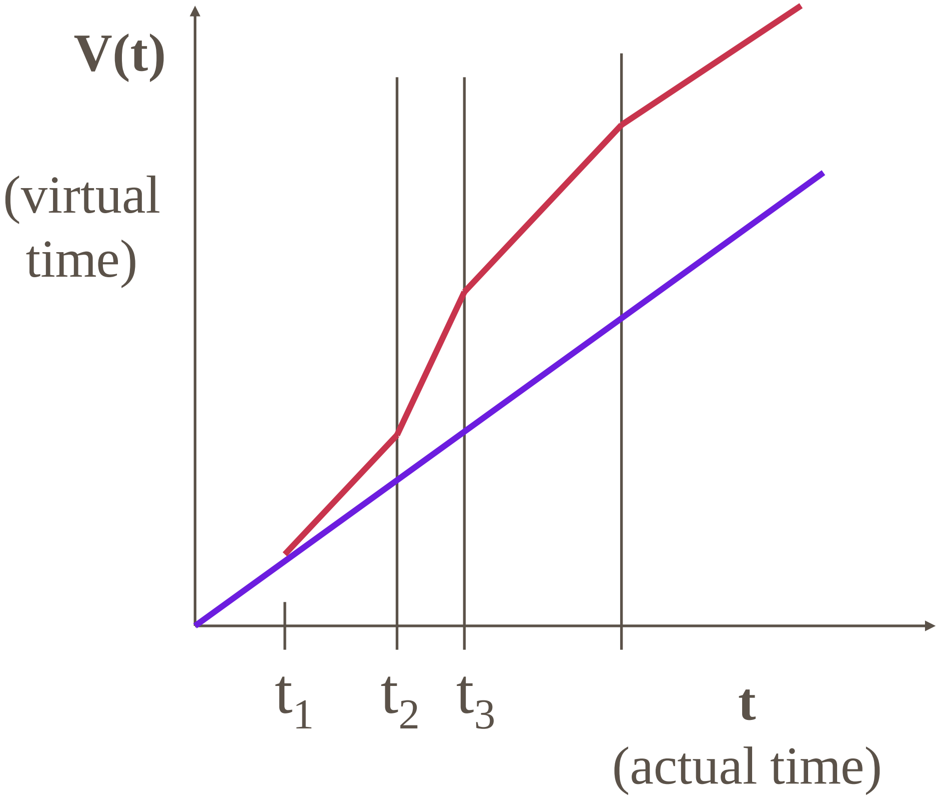
PGPS: Virtual clock vs. Actual clock (1/5)

- The curve function is changing from one interval to another
- An interval is (t_{j-1}, t_j), j=2, 3, …
- The slope is $$\frac{1}{\sum_{j \in B_{GPS}} \phi_j}$$
- Consider a **busy period** that begins at time zero
- V(t) evolves as follows:
  - V(0) = 0
  - $V(t_{j-1}+\tau) = V(t_{j-1}) + \tau / \sum \phi_j$, $\tau \le t_j - t_{j-1}$, j=2,3, …



PGPS: Virtual clock vs. Actual clock (2/5)

- V(t) changes at the **rate** of **1**/$\sum\phi_j$
- For a backlogged session
  - When $\sum \phi_j < 1$, it seems that the "corresponding" server becomes **faster**, from ϕ_i to $\phi_i * 1/\sum \phi_j$
- For **individual** backlogged sessions, the portion of service rate received ***increases***.



PGPS: Virtual clock vs. Actual clock (3/5)

- $1/\sum \phi_j$ represents the "current" service rate *from the **backlogged** sessions' point of view*
- Each backlogged session receives service at **rate $\phi_i * (\partial V(t_j+\tau) / \partial \tau)$**
- V(t) is a non-decreasing function
- Packet service order of a session is FIFO

PGPS: Virtual clock vs. Actual clock (4/5)

- $a_{i,k}$: the (**actual) arrival time** of the k^{th} packet of session *i*
- $V(a_{i,k})$: the virtual time of $a_{i,k}$
- Need to obtain a correspondence of a ***packet arrival time*** and *departure time* in the virtual time domain
- $\mathbf{S_{i,k}}$: the virtual time that packet *k* of session *i* begins its **service**
- $\mathbf{F_{i,k}}$: the **virtual finishing time** of $a_{i,k}$
- We have
  - $\mathbf{S_{i,k} = \max\{F_{i,k-1}, V(a_{i,k})\}}$
  - $\mathbf{F_{i,k} = S_{i,k} + L_{i,k} / \phi_i}$

where $L_{i,k}$ is the packet length
and $L_{i,k} / \phi_i$ is the "presumed" service time, i.e. the ***worst-case*** **service time**

PGPS: Virtual Finishing Time (5/5)

- When a packet arrives, virtual clock is updated and the packet is stamped with its **virtual finishing time** **($F_{i,k} = \max\{F_{i,k-1}, V(a_{i,k})\} + L_{i,k} / \phi_i$)**

- Server is work-conserving and serves packets in *an* ***increasing order of virtual finishing time***
- Virtual times are updated when an arrival or departure occurs (***rate change***).
- The system must keep track of the set of $B_j(t)$ (***the set of backlogged sessions at time t***).

Relationship between a fluid GPS and its corresponding WFQ systems

- In terms of queueing delay, a packet will finish its service in a WFQ system **later** than in the GPS system by **NO more than** the transmission time of **one** maximum size packet
- In terms of total number of bits served for a session, a WFQ system **does NOT fall behind** a corresponding GPS system by **more than one** maximum size packet.

Summary

- Fair queueing to support QoS
- WFQ
  - Approximating GPS
  - **Minimum throughput guarantee**
  - **Flow isolation**
  - **Delay bound guarantee**
  - **Weighted Fairness**

Other Scheduling Algorithms

- WF2Q
- WF2Q-M
- Deficit Round Robin (DRR)
- etc.

References

- A. K. Parekh and R. G. Gallager, "*A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case*," IEEE/ACM Transactions on Networking, Vol. 1, No. 3, pp.344-357, June 1993.
- S. Golestani, "A Self-clocked Fair Queueing Scheme for Broadband Applications," In Proceedings of IEEE INFOCOM'94, page 636-646, Toronto, CA, Jane 1994.
- J. C. R. Bennett and H. Zhang, "WF2Q: Worst-case Fair Weighted Fair Queueing," in Proc. IEEE INFOCOM'96, San Francisco, CA, Mar. 1996.
- M. Shreedhar and George Varghese, "Efficient Fair Queuing using Deficit Round Robin*,"* ACM SIGCOMM 1995.
- Jeng Farn Lee, Meng Chang Chen and Yeali S. Sun, "WF2Q-M: Worst-case Fair Weighted Fair Queueing with Maximum Rate Control," Computer Networks, Volume 51, Issue 6, pp. 1403-1420, April 2007.
- S. Floyd and V. Jacobson, "Link-sharing and Resource Management Models for Packet Networks," IEEE/ACM Trans. Networking, vol. 3 pp. 365-386, Aug. 1995.
- H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Network," Proc. IEEE, Vol. 83, October 1995, pp. 1374-1396.
- J. C. R. Bennett and H. Zhang, "Hierarchical Packet Fair Queueing Algorithms," IEEE/ACM Trans. Networking, vol. 5, pp. 675-689, Oct. 1997.
- Kurose Chapter 4.