Scheduling (1)

Area de Ingeniería Telemática
http://www.tlm.unavarra.es

Redes
4º Ingeniería Informática

Temario

1. Introducción a las redes
2. Encaminamiento
3. Transporte extremo a extremo
4. **Arquitectura de conmutadores de paquetes**
5. Tecnologías para redes de área local
6. Tecnologías para redes de área extensa y última milla
7. Conmutación de circuitos

Resumen de lo visto

- Arquitectura de routers con plano de control y plano de datos
- Problemas que deben resolverse para construir el plano de datos
  - Address lookup (prefix-trees)
  - Acceso a memoria y cálculo para manipulación de cabeceras
  - Conmutación de paquetes en el plano de datos
  - Switching fabrics: buses, crossbar, broadcast, banyan...
  - Colas a la entrada o a la salida
  - El problema del scheduling

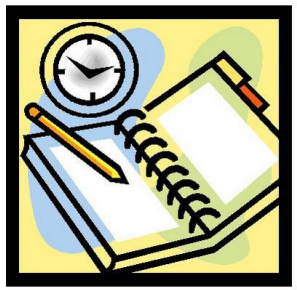
- A continuación: scheduling

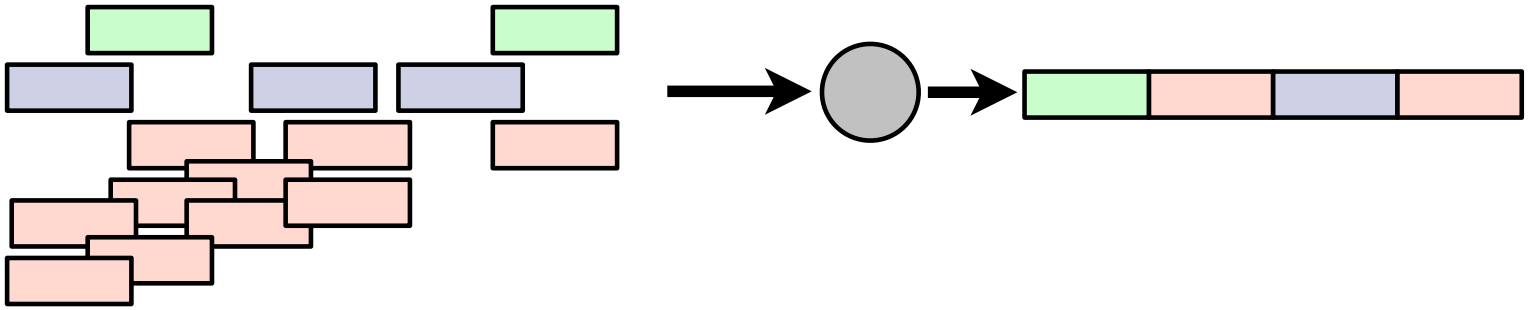
Objetivos

- Conocer los principios y características de la planificación en redes

Scheduling

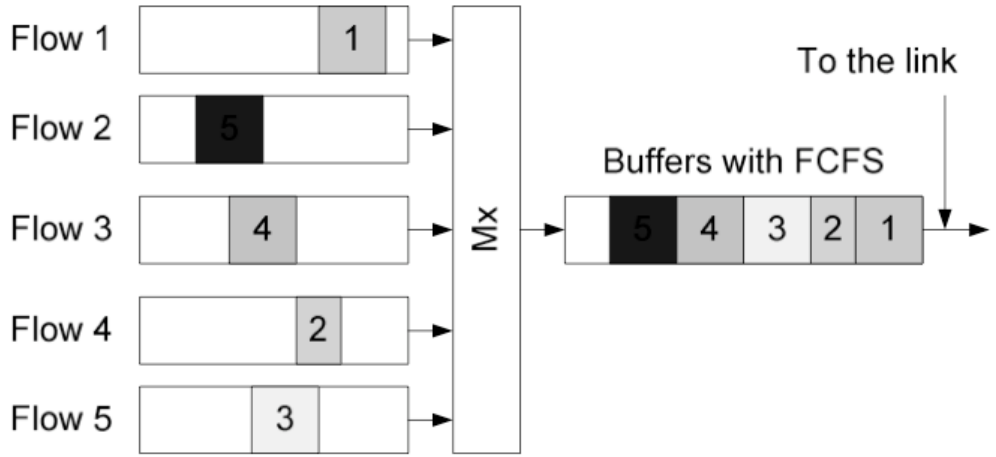
- Permite compartir recursos
- Emplea una disciplina de planificación para decidir la siguiente petición a atender
- Puede tener lugar en diferentes niveles de una pila de protocolos
- Nos centraremos en compartir la capacidad de un enlace
- Por ejemplo en el nivel de aplicación sería necesario para decidir la siguiente petición a un servidor que atender
- Nos centraremos en planificadores conservativos en trabajo (*work conserving*): están inactivos solo si la cola está vacía





FCFS (FIFO)

- Orden de llegada
- Almacenamiento y reenvío
- Es el método más rápido y sencillo de implementar
- Se suele utilizar por defecto (B*est Effort*)
- Limitado por la capacidad del buffer ante congestión (normalmente en número de paquetes)
- No permite diferenciar entre distintos tipos de paquete
- Se logra asignación proporcional a la demanda
- Una fuente *greedy* puede capturar el enlace



The Conservation Law

- La disciplina FCFS no distingue entre diferentes flujos
- FCFS por ejemplo no permite menor retardo a paquetes de un flujo

Conservation Law

- Nos dice que una disciplina de planificación solo puede mejorar el retardo medio de un flujo frente a FCFS a costa de empeorar el de otro flujo
- Sea un conjunto de N flujos en un planificador
- Para el flujo *i* la tasa media de llegadas por unidad de tiempo es λ_i
- El tiempo medio de servicio de los paquetes del flujo *i* es x_i
- La utilización media del enlace debido al flujo *i* es $\rho_i = \lambda_i x_i$
- El tiempo medio de espera en cola de los paquetes del flujo *i* es q_i
- Si el planificador es conservativo en trabajo (*work-conserving*) entonces

$$\sum_{i=1}^{N} \rho_i q_i = \text{Constante}$$

- Es independiente del planificador en concreto
- Es decir: para reducir el retardo medio de una clase debemos aumentar el de otra(s)

The Conservation Law

- Enlace a 155 Mbps, paquetes de 53 bytes
- Dos flujos
  A. Tasa de llegadas de 10Mbps
  B. Tasa de llegadas de 25Mbps
- Con FCFS ambos sufren un retardo medio en cola de 0.5 ms
- Con un planificador diferente los paquetes del flujo A sufren un retardo medio en cola de 0.1 ms
- ¿Cuál es el retardo medio en cola que sufren los paquetes del flujo B?

- Todos los paquetes de igual tamaño así que el tiempo de servicio no afecta

$$\sum_{i=1}^{N} \rho_i q_i = \text{Constante}$$

$$10 / 155 \times 0.5 + 25 / 155 \times 0.5 = 10 / 155 \times 0.1 + 25 / 155 \times R_B$$

$$R_B = 0.66 \text{ ms}$$

Características deseables

- **Sencillo de implementar**
- **Reparto justo y protección**
- **Performance bounds (deterministas o estadísticos)**
- **Que permita implementar un CAC simple**

Características deseables

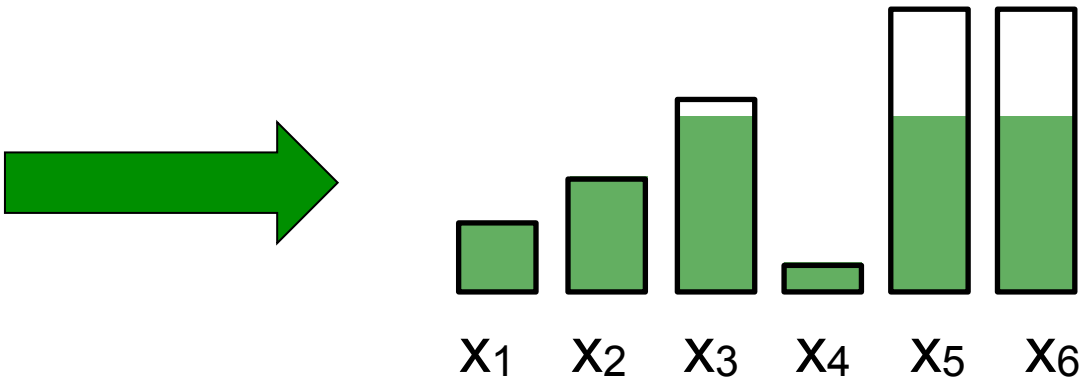
- **Sencillo de implementar**
  - Que requiera pocas operaciones para ser rápido, implementable en hardware
  - Que el número de operaciones sea independiente del número de flujos a planificar

Características deseables

- **Sencillo de implementar**
- **Reparto justo y protección**
  - Flujos con requerimientos estrictos deben tenerlos garantizados independiente de esta "justicia"
  - Reparto justo es importante para flujos best-effort
  - Reparto decimos que es justo si satisface un *max-min fair share*
  - Scheduling es normalmente una decisión local al nodo de conmutación pero la "justicia" para un flujo es un objetivo global
  - Un flujo debería enviar la menor tasa de todas sus asignaciones justas en el trayecto de origen a destino
  - Lograr justicia global con flujos cambiantes no es tan sencillo
  - La protección implica que un flujo que envíe más que su asignación justa no afecte al resto
  - Un planificador que haga un reparto justo ofrece protección
  - (...)

Reparto justo (max-min fair)

- Para dividir recursos entre un conjunto de usuarios, todos con iguales "derechos" pero diferentes demandas
- De forma simple: a los que piden "poco" se les da lo que piden y lo que sobra se reparte entre los que piden "mucho"
- Formalmente:
  - Asignar recursos en orden creciente de demanda
  - Ningún cliente recibe más de lo que solicita
  - Aquellos cuya demanda no se pueda satisfacer se reparten el remanente del recurso



Reparto justo (max-min fair)

- Flujos 1,...,n
- Demandas $x_1,...,x_n$
- Demandas ordenadas $x_1 \leq x_2 \leq ... \leq x_n$
- Capacidad a repartir C
- Inicialmente asignar C/n al flujo 1
- Si esto es más que lo que necesita ($C/n > x_1$) lo que sobra se repartirá entre el resto
- Asignar al flujo 2: C/n más la parte que le corresponde de lo que sobró del flujo 1, es decir:

$$C/n + \frac{C/n - x_1}{n-1}$$

- Esto puede ser más que lo que el flujo 2 necesita, así que lo que sobra se puede repartir entre el resto
- Al final todos tienen lo que han pedido o si no había suficiente para eso no tienen menos que cualquier otra fuente que ha pedido más

Max-min Fair (Ejemplo)

- Recurso: 10
- Demandas: 2, 2.6, 4 y 5
- 10/4 = 2.5
  - Demasiado para el primer cliente
  - Asignarle 2 y queda 0.5
- Ese 0.5 repartirlo entre los otros 3:
  - 0.5/3 = 0.16666…
  - Asignaciones [2, 2.66, 2.66, 2.66]
  - Demasiado para el segundo cliente
  - Asignarle 2.6 y quedan 0.0666….
- Repartir ese 0.0666… entre los otros 2:
  - (2.5+0.5/3-2.6)/2 = 0.03333…
  - Asignaciones [2, 2.6, 2.7, 2.7]

Max-min Weighted Fair Share

- En ocasiones se desea una asignación preferente a unos flujos frente a otros
- Se asocia esta preferencia con unos pesos w_1, w_2, ..., w_n
- Extensión:
  - Los recursos se asignan en orden de demanda creciente, normalizada por el peso
  - Ningún cliente recibe más de lo que solicita
  - Aquellos cuya demanda no se pueda satisfacer se reparten el remanente del recurso en proporción a sus pesos

Max-min WFS (Ejemplo)

- Recurso: 20. Demandas: 4, 2, 10 y 8. Pesos: 2.5, 4, 0.5 y 1
- Normalización de los pesos:
  - Que el menor valga 1
  - Pesos normalizados: 5, 8, 1 y 2
- En vez de 4 clientes es como si hubiera 5+8+1+2 = 16
- C/n = 20/16 = 1.25
- La asignación a cada uno sería:
  - (5x1.25=) 6.25, (8x1.25=) 10, (1x1.25=) 1.25 y (2x1.25=) 2.5
  - El cliente 1 obtiene 6.25 pero solicitaba 4 luego sobra 2.25
  - El cliente 2 obtiene 10 pero solicitaba 2 luego sobra 8
  - El cliente 3 obtiene 1.25 pero solicita 10 (insuficiente)
  - El cliente 4 obtiene 2.5 pero solicita 8 (insuficiente)
- Ha sobrado 2.25 + 8 = 10.25 a repartir entre los clientes 3 y 4
  - Sus pesos ya están normalizados (1 y 2). C/n = 10.25 / 3 = 3.417
  - El cliente 3 obtiene 3.417 adicional, en total 1.25+3.417 = 4.667 (insuficiente)
  - El cliente 4 obtiene 6.834 adicional, en total 2.5+6.834 = 9.334, sobra 1.334
  - Lo que sobra del cliente 4 se asigna al cliente 3 y así recibe 4.667+1.334
- Asignación final: 4, 2, 6, 8

Características deseables

- **Sencillo de implementar**
- **Reparto justo y protección**
- **Performance bounds**
  - Debería permitir garantizar límites (*bounds*) a un flujo
  - Bounds extremo a extremo, lo cual implica a todos los schedulers en el camino
  - Más simple si emplean la misma disciplina de planificación
  - (...)

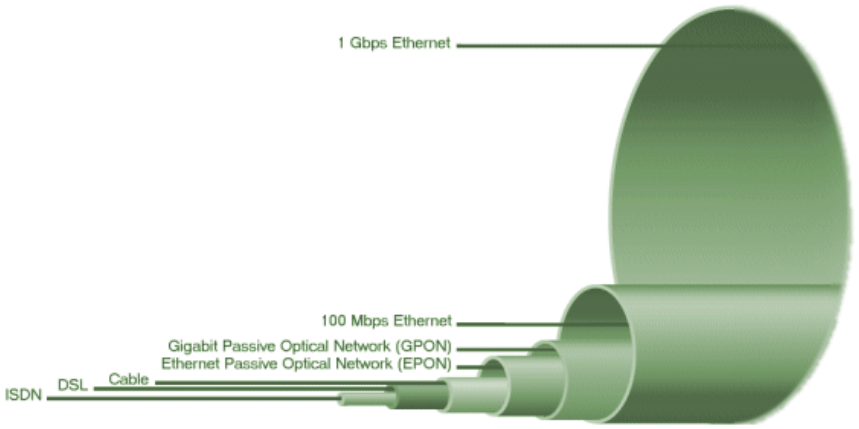
Performance bounds

- **Deterministas**
  - Se cumplen para todos los paquetes del flujo
  - Ejemplo: 10s como cota máxima al retardo extremo a extremo implica que todos los paquetes sufrirán un retardo menor que 10s
- **o Estadísticos**
  - Probabilístico
  - Ejemplo: un máximo de 10s de retardo con probabiliadad 0.99 implica que la probabilidad de que un paquete sufra más de 10s de retardo es menor del 1%
  - Otra forma es en forma de “uno entre N”
  - Ejemplo: No más de 1 entre 100 paquetes sufrirán un retardo de más de 10s
  - Este caso es más fácil de verificar pero más difícil de implementar pues los conmutadores deben seguir el estado de cada flujo
- Los deterministas suelen requerir más recursos pues son para todos los paquetes

Performance parameters

Bandwidth

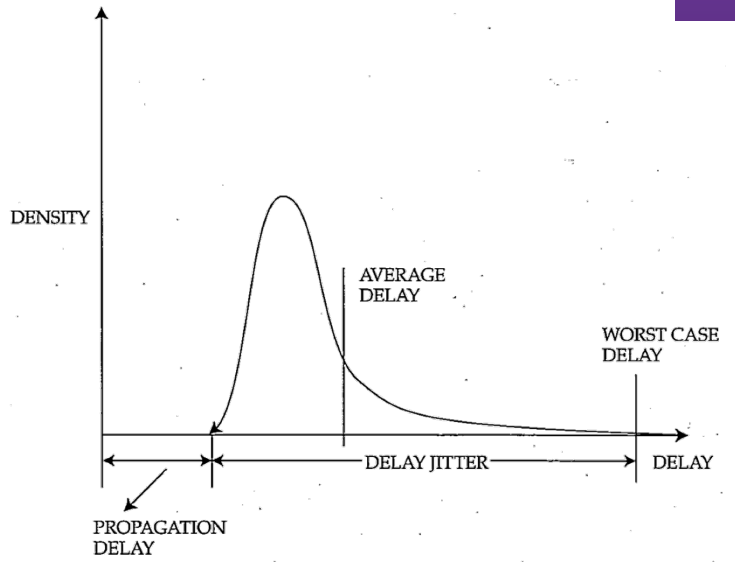
- Recibir al menos un mínimo, medido en un intervalo
- Es el más habitual en las implementaciones
- (...)



Performance parameters

Delay

- Determinista o estadístico
- Caso peor: suponer que el resto de flujos se comportan de la peor manera posible
- Medio:
  - Debe ser la media para cualquier patrón de llegadas
  - O sea, imposible de medir
  - Eso lo hace difícil de garantizar
  - Normalmente hablaremos de la media en la duración de un flujo
  - Suele ser una aproximación si dura suficiente y el flujo es independiente del resto
- (...)



Performance parameters

Delay

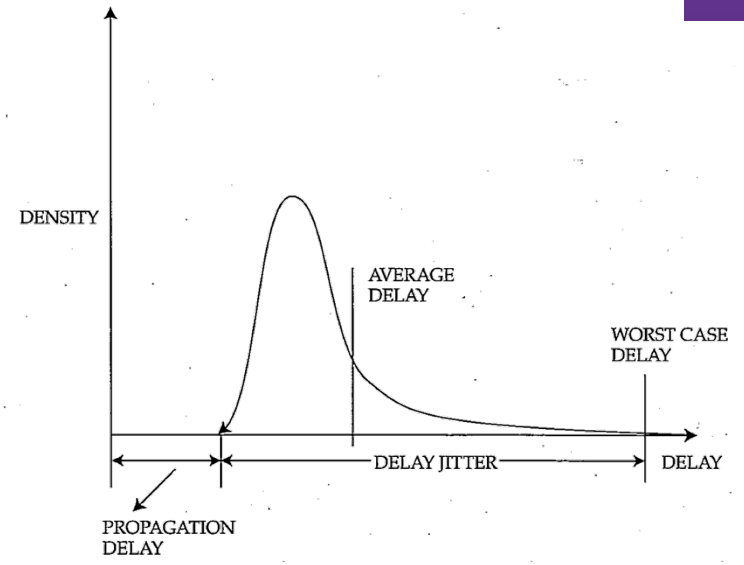
- Percentil:
  - Por ejemplo que el 99% de los paquetes sufran menos de un retardo
  - De nuevo medible solo para un flujo en concreto
  - Algunos schedulers tienen una dependencia entre BW y delay de forma que para lograr bajo delay hace falta alto BW

Delay-jitter

- Acotar la diferencia entre el mayor y el menos retardo posible

Losses

- Fracción de paquetes (del flujo)



Características deseables

- **Sencillo de implementar**
- **Reparto justo y protección**
- **Performance bounds (deterministas o estadísticos)**
- **Que permita implementar un CAC simple**
  - Dado el conjunto de flujos existentes y sus requisitos y el nuevo flujo y los suyos
  - Decidir si se pueden alcanzar los requisitos del nuevo flujo sin violar los de los anteiores
  - Hacerlo sin infrautilizar la red (no aceptando a penas flujos será fácil cumplir los objetivos pero infrautilizamos la red)

Resumen

- FCFS
- The conservation law
- Max-min fairness
- Performance bounds
- Performance parameters