Algoritmos de Escalonamento

FEUP/DEEC – 2001/02
José Ruela

Escalonamento - necessidade

» A partilha de recursos em redes de comunicação e, em particular, a adopção de estratégias de multiplexagem estatística, originam situações de contenção, devido à competição pela utilização de recursos

» Em redes que suportam integração de serviços torna-se necessário providenciar QoS diferenciada por categorias de serviço (ou classes de tráfego), oferecendo garantias de desempenho a aplicações críticas e ao mesmo tempo permitindo uma partilha de recursos de acordo com critérios de equidade (*fairness*)

» Os algoritmos de escalonamento (*scheduling algorithms*) são um componente essencial para atingir estes objectivos, uma vez que determinam as disciplinas de serviço a aplicar a fluxos aceites pela rede

» Um algoritmo de escalonamento realiza duas funções

- decide a ordem de serviço de fluxos em competição (pedidos de serviço)
- gere as filas de pedidos de serviço

Escalonamento - objectivos

» Os algoritmos de escalonamento são usados em qualquer sistema ou camada protocolar em que ocorra contenção por recursos, mas em especial em elementos de rede (*routers*, comutadores, etc.)

» Desempenham um papel importante na provisão de diferentes níveis de QoS a diferentes aplicações, permitindo controlo diferenciado de atraso, largura de banda ou taxa de perdas

- aplicações com requisitos de tempo real necessitam de garantias absolutas
- aplicações *best-effort* não necessitam de quaisquer garantias, mas é desejável que a partilha de recursos entre estas aplicações seja feita de forma equitativa (*fair*)

» Em geral pretende-se que os algoritmos de escalonamento sejam simples (fáceis de implementar), tratem de forma equitativa tráfego *best-effort* e garantam ao restante tráfego os níveis de desempenho negociados e facilitem os mecanismos associados de controlo de admissão de conexões

Fairness

» A atribuição equitativa de recursos é habitualmente baseada no critério designado por *max-min fairness*, que atribui recursos do seguinte modo:

- Os recursos são atribuídos por ordem ascendente dos valores solicitados
- Nenhum fluxo obtém uma quota de utilização superior ao solicitado
- Fluxos cujos requisitos não podem ser satisfeitos são tratados de forma idêntica (recebem a mesma quota de utilização – *fair share*)

» Exemplo (capacidade a partilhar: 15)

- Pedidos: 1, 2, 5, 8, 10
- Atribuição: 1, 2, 4, 4, 4

» O objectivo de partilha equitativa é global, enquanto que os algoritmos de escalonamento têm carácter local; assim, os recursos atribuídos a uma fluxo correspondem ao menor valor atribuído ao longo do percurso

» O critério *max-min fairness* oferece ainda um mecanismo de protecção relativamente a fontes que tentam transmitir acima da respectiva quota

Opções

» Na concepção de um mecanismo de escalonamento é possível considerar quatro graus de liberdade (opções)

– Número de níveis de prioridade

– Disciplina *work-conserving* ou *non-work-conserving* em cada nível

– Grau de agregação de fluxos em cada nível

– Ordem de serviço em cada nível

Prioridade

» A cada conexão é atribuído um nível de prioridade

» Só são servidos pacotes num determinado nível de prioridade se não existirem pacotes à espera de ser servidos em qualquer nível com prioridade mais alta

» Quanto mais elevado for o nível de prioridade mais baixo é o atraso médio sofrido pelos pacotes

» Uma possível consequência desta estratégia é a negação de serviço (*starvation*) a pacotes nos níveis de prioridade mais baixos, o que pode ser combatido com políticas de controlo de admissão (limitando a capacidade disponível para cada nível, excepto o mais baixo) e de policiamento

Work-conserving versus non-work-conserving

» Em disciplinas *work-conserving* um servidor está inactivo (*idle*) apenas quando não existe qualquer pacote para transmitir

» Em disciplinas *non-work-conserving* um servidor pode estar inactivo, mesmo que exista(m) pacote(s) para transmitir

- a cada pacote é atribuído um instante em que é elegível para transmissão, com o objectivo de controlar o débito (garantindo conformidade com um dado descritor) ou a variação do atraso (de modo a compensar o *jitter* introduzido no passo anterior)
- se um pacote não for elegível, não será transmitido (será atrasado até ser elegível), mesmo que o servidor esteja livre

» Disciplinas *non-work-conserving* aumentam o atraso médio dos pacotes, desperdiçam largura de banda (que pode no entanto ser reutilizada por tráfego *best-effort*) e penalizam *sempre* fontes que excedem o débito contratado (mesmo que haja recursos livres)

Grau de agregação de fluxos

» É possível considerar vários graus de agregação de fluxos

» Num extremo, não existe qualquer agregação, isto é, os fluxos são tratados individualmente, o que permite diferenciar a QoS por fluxo mas requer a manutenção de informação de estado por ligação e portanto implica uma maior complexidade do escalonador

» No outro extremo a agregação é total, usando-se uma única variável de estado para descrever todas as ligações, que partilham a mesma QoS (o escalonador é mais simples, mas a diferenciação é impossível)

» Uma solução intermédia consiste em agrupar fluxos em classes; é necessário manter informação de estado por classe e os pacotes da mesma classe recebem tratamento idêntico (a diferenciação de QoS é feita por classe)

» Para algumas disciplinas de serviço, quanto maior for o grau de agregação menor é o número de ligações que podem ser aceites

- Para garantir limites de desempenho negociados a qualquer fluxo numa classe (por exemplo, o atraso máximo), a impossibilidade de diferenciar e controlar fluxos individuais obriga a que o algoritmo de admissão de conexões considere o caso mais desfavorável (por exemplo, a ocorrência de *bursts* simultâneos dos fluxos)

Ordem de Serviço

» É necessário decidir a ordem pela qual devem ser servidos, em cada nível de prioridade, os pacotes de fluxos individuais ou agregados

» A estratégia mais simples consiste em servir os pacotes pela ordem de chegada (FCFS – *First Come First Serve*); esta estratégia não permite oferecer garantias quanto a atrasos nem protecção relativamente a fluxos que tentem monopolizar o uso da largura de banda, isto é, a atribuição de recursos não obedece ao critério *max-min fairness*

» A alternativa consiste em servir os pacotes por uma ordem diferente da ordem de chegada, de acordo com uma etiqueta de serviço (*service tag*) associada a cada pacote

– as propriedades de um algoritmo de escalonamento e as garantias de desempenho associadas (débito, atraso, etc.) dependem da forma como são calculadas as etiquetas

– qualquer disciplina de serviço que não seja FCFS implica um *overhead* acrescido

Exemplos de Algoritmos de Escalonamento

» *Work-conserving*

– *Generalized Processor Sharing* (GPS) / *Fluid Fair Queueing* (FFQ) (ideais)

– *Round-Robin* (RR) e *Weighted Round-Robin* (WRR)

– *Fair Queueing* (FQ), *Weighted Fair Queueing* (WFQ), *Worst-case Fair Weighted Fair Queueing* (WF^2Q) e *Self-Clocked Fair Queueing* (SCFQ)

– *Virtual Clock* (VC)

– *Delay Earliest-Due-Date* (D-EDD)

» *Non-work-conserving*

– *Stop-and-Go* (SG)

– *Hierarchical Round-Robin* (HRR)

– *Jitter Earliest-Due-Date* (J-EDD)

– *Rate-Controlled Static Priority* (RCSP)

GPS – Generalized Processor Sharing

» O objectivo deste algoritmo é a atribuição de recursos de acordo com o critério *max-min fairness*

» GPS é uma disciplina *work-conserving* ideal (não implementável)

» Existe uma fila por ligação (fluxo)

– Durante qualquer intervalo de tempo em que existam N filas não vazias, o servidor serve *simultaneamente* os N pacotes à cabeça das filas, cada um com um débito igual a 1/N da capacidade da ligação física (N varia à medida que se completam e/ou iniciam serviços)

– De forma equivalente se pode dizer que é servida uma quantidade de dados *infinitesimal* em cada visita a uma fila (uma aproximação consistiria em servir rotativamente um bit de cada fila não vazia)

» Nenhuma disciplina realizável pode ser tão justa (no sentido *max-min*) quanto GPS, pois enquanto um pacote é servido, existe injustiça para os restantes; o grau de injustiça (*unfairness*) pode ser limitado

» É possível associar pesos às ligações; as ligações recebem serviço proporcionalmente aos pesos sempre que a fila não esteja vazia

GPS – Generalized Processor Sharing

» Uma ligação diz-se *backlogged* sempre que existam dados na sua fila

» Considerando N ligações com pesos associados ϕ_i e que partilham um canal com capacidade C, em qualquer instante τ o débito de uma ligação é $C\frac{\phi_i}{\sum_{j\in B(\tau)}\phi_j}$ sendo B(τ) o número de filas não vazias nesse instante

» Seja $S(i, t_1, t_2)$ a quantidade de dados da ligação *i* servida no intervalo $[t_1, t_2]$. Para uma ligação *i* que esteja *backlogged* no intervalo $[t_1, t_2]$ e qualquer outra ligação *j*, verifica-se a relação

$$\frac{S(i,t_1,t_2)}{S(j,t_1,t_2)} \geq \frac{\phi_i}{\phi_j}$$

– Uma ligação não *backlogged* recebe o serviço de que necessita; ligações *backlogged* (recebem menos serviço do que o pretendido) partilham a capacidade remanescente proporcionalmente aos seus pesos

» A designação FFQ (*Fluid Fair Queueing*) é também usada como sinónimo de GPS

Emulação de GPS

» A disciplina GPS (ou FFQ) não é implementável

» Várias disciplinas de serviço foram propostas com o objectivo de emular GPS

» A emulação mais simples de GPS é a disciplina *Round-Robin* (RR), que admite uma variante com pesos associados às ligações – *Weighted Round-Robin* (WRR)

» As disciplinas *Weighted Fair Queueing* (WFQ), também designada por *Packetized Generalized Processor Sharing* (PGPS) e *Worst-case Fair Weighted Fair Queueing* (WF^2Q) são emulações mais elaboradas de GPS

» A complexidade computacional de WFQ e WF^2Q motivou a proposta de uma disciplina de serviço mais simples, designada *Self-Clocked Fair Queueing* (SCFQ)

WRR - Weighted Round-Robin

» *Round-Robin* serve rotativamente um *pacote* de cada fila não vazia (em vez de realizar serviço *infinitesimal*, como em GPS)

– Aproxima razoavelmente GPS se os pacotes tiverem comprimento fixo e se as ligações tiverem o mesmo peso; não é justo se os pacotes tiverem diferentes comprimentos e os pesos não forem iguais

» WRR serve as ligações proporcionalmente aos pesos atribuídos

– Caso 1: pesos diferentes e pacotes com comprimento fixo; é servido mais do que um pacote por fila e por ciclo, após normalização para obter pesos inteiros

– Caso 2: pesos diferentes e pacotes com comprimento variável; os pesos são normalizados pelo tamanho médio dos pacotes, como exemplificado a seguir:

• pesos {0.5, 0.75, 1.0}, tamanho médio dos pacotes {50, 500, 1500}

• pesos normalizados: {0.5/50, 0.75/500, 1.0/1500} = {60, 9, 4}

WRR - Weighted Round-Robin

» O algoritmo WRR pressupõe o conhecimento do tamanho médio dos pacotes gerados por cada fonte, o que em muitos casos é imprevisível e impede uma distribuição de recursos de acordo com o critério *max-min fairness*

» O algoritmo é justo apenas em escalas temporais superiores à duração de um ciclo, podendo ser injusto durante ciclos longos para ligações com pequeno peso ou quando existe um número elevado de ligações, como se exemplifica:

- Ligação T3 (45 Mbit/s) com 500 ligações (250 com peso 1 e 250 com peso 10); tamanho médio dos pacotes: 500 *bytes*
- Tempo de transmissão de um pacote: 500 * 8 / 45 = 88.8 μs
- Duração de um ciclo: (250 * 10 + 250 *1) * 88.8 = 244.2 ms
- Em intervalos muito menores do que 244.2 ms, algumas ligações obtêm uma quota de serviço muito superior à de outras

WFQ - Weighted Fair Queueing

» O algoritmo WFQ não assume tamanhos infinitesimais dos pacotes (como GPS) nem requer conhecimento do tamanho médio dos pacotes (como WRR)

» Em WFQ, quando o servidor está pronto a transmitir um novo pacote no instante τ, selecciona entre os pacotes à espera nesse instante o primeiro que completaria o seu serviço num servidor GPS de referência, admitindo que nenhum outro pacote chegava depois de τ (em GPS o próximo pacote a escalonar poderia ainda não estar presente no sistema no instante τ)

» Uma vez que WFQ apenas tem em conta os tempos de conclusão do serviço no servidor GPS de referência, considera elegíveis todos os pacotes presentes no instante τ em que um novo serviço se pode iniciar, incluindo os que em GPS apenas iniciariam o seu serviço mais tarde, e que assim podem beneficiar da conclusão do serviço corrente antes do que ocorreria em GPS

WF²Q – Worst-case Fair Weighted Fair Queueing

» O algoritmo WF²Q considera, para além dos tempos de conclusão de serviço no servidor GPS, os tempos de início de serviço

» Quando selecciona um pacote para início de um novo serviço no instante τ, o servidor considera elegíveis apenas os pacotes que nesse instante já tinham começado (e eventualmente acabado) o respectivo serviço no servidor GPS; será seleccionado (como em WFQ) o pacote com menor tempo de conclusão de serviço GPS

» Em WFQ e WF²Q o serviço acumulado por ligação nunca se atrasa, em relação a GPS, mais do que o tamanho do maior pacote

– WFQ e WF²Q apresentam o mesmo limite superior para o atraso

» WFQ pode adiantar-se significativamente em relação a GPS (o que não acontece com WF²Q, em que o avanço ou o atraso não excedem o tamanho de um pacote)

WFQ e WF²Q – algoritmo

» Exceptuando a condição de elegibilidade dos pacotes, os algoritmos são semelhantes e baseiam-se na actualização de uma variável de estado $F_{i,j}^{k}$ (*Virtual Finish Time*), associada a cada pacote *k*, no servidor *i* e na ligação *j*

$$F_{i,j}^{k} \leftarrow \max\{V_i(a_{i,j}^{k}), F_{i,j}^{k-1}\} + \frac{L_j^{k}}{\phi_{i,j}}$$

sendo

– $V(t)$ - *Virtual time* do sistema (medida do progresso do serviço)

– L_j^{k} - Tamanho do pacote (bits)

– $a_{i,j}^{k}$ - Instante de chegada do pacote

» A função *V* é definida em intervalos $[t_1, t_2]$ em que se realiza serviço por

$$V(t_1) = 0 \quad e \quad \frac{\partial V(\tau)}{\partial \tau} = \frac{1}{\sum_{j \in B(\tau)} \phi_j} \qquad \forall t_1 \le \tau \le t_2$$

sendo $B(\tau)$ o número de ligações *backlogged* no sistema PGS de referência

WFQ e WF^{2}Q - exemplo

» Assume-se que o tamanho dos pacotes é idêntico e normalizado (valor igual a 1) e que a velocidade de transmissão no canal partilhado é 1

» Consideram-se 11 ligações com as seguintes características

– Ligação 1

◆ Débito a garantir = 0.5

◆ Gera 11 pacotes contíguos, com início nos instantes t = k (k = 0, ..., 10)

– Ligações 2 a 11 (idênticas)

◆ Débito a garantir = 0.05

◆ Geram 1 pacote com início em t = 0

» Instantes de início e conclusão de serviço no servidor GPS de referência

– Ligação 1

◆ Início dos serviços - 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20

◆ Conclusão dos serviços - 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 21

– Ligações 2 a 11

◆ Início dos serviços - 0

◆ Conclusão dos serviços - 20

WFQ e WF^{2}Q - exemplo

L1

L2 - L11

Serviço WFQ

Serviço WF^{2}Q

SCFQ - Self-Clocked Fair Queueing

» A complexidade computacional do cálculo e actualização da variável *Virtual Finish Time* usada em WFQ e WF^2Q para emular um servidor GPS está na origem do algoritmo SCFQ (*Self-Clocked Fair Queueing*) que permite estimar de forma aproximada o valor dessa variável

» A aproximação consiste em substituir o valor corrente da função $V(t)$ pelo instante em que o pacote actualmente a ser servido no sistema SCFQ completa o serviço

» Este algoritmo, embora simples, pode provocar situações de injustiça relativa em intervalos de tempo curtos, podendo causar atrasos no serviço muito superiores aos que ocorrem em WFQ

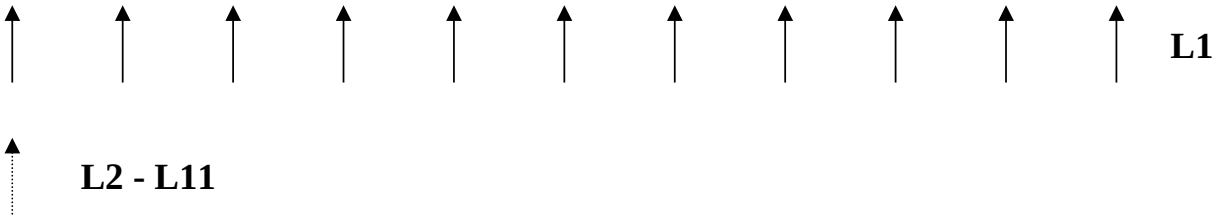
WFQ e SCFQ - exemplo

» Assume-se que o tamanho dos pacotes é idêntico e normalizado (valor igual a 1) e que a velocidade de transmissão no canal partilhado é 1

» Consideram-se 11 ligações com as seguintes características
  – Ligação 1
    ◆ Débito a garantir = 0.5
    ◆ Gera 11 pacotes com início nos instantes t = 2k (k = 0, ..., 10)
  – Ligações 2 a 11 (idênticas)
    ◆ Débito a garantir = 0.05
    ◆ Geram 1 pacote com início em t = 0

» Instantes de início e conclusão de serviço no servidor GPS de referência
  – Ligação 1
    ◆ Início dos serviços - 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20
    ◆ Conclusão dos serviços - 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 21
  – Ligações 2 a 11
    ◆ Início dos serviços - 0
    ◆ Conclusão dos serviços - 20

WFQ e SCFQ - exemplo



Serviço WFQ

Serviço SCFQ

VC - Virtual Clock

» O algoritmo *Virtual Clock* tem como objectivo emular um sistema TDM (*Time Division Multiplexing*)

» A cada pacote é atribuído um instante virtual de transmissão (*virtual transmission time*), correspondente a um serviço TDM; os pacotes são servidos por ordem crescente desses tempos

» O algoritmo baseia-se na actualização da variável de estado *auxVC* (*auxiliary Virtual Clock*)

$$auxVC_{i,j}^{k} \leftarrow \max\{a_{i,j}^{k}, auxVC_{i,j}^{k}\} + Vtick_{i,j}$$

sendo

– $Vtick_{i,j}$ - intervalo médio entre chegadas de pacotes na ligação *j*