Introduction aux systèmes temps réel

Département informatique et réseaux

Frank Singhoff

C-214

singhoff@enst.fr

Sommaire

1. Principes généraux sur les systèmes temps réel.
2. Exemple : les applications multimédias réparties.
3. Conclusion, synthèse et perspectives.
4. Références.

Partie 1

Principes généraux sur les systèmes temps réel

Sommaire

Sous partie 1.1

Présentation, définitions

Présentation, définitions (1)

- "*En informatique temps réel, le comportement correct d'un système dépend, non seulement des résultats logiques des traitements, mais aussi du temps auquel les résultats sont produits*" [STA 88].
- Objectifs :
  - Déterminisme logique : les mêmes entrées appliquées au système produisent les mêmes résultats.
  - Déterminisme temporel : respect des contraintes temporelles (temps de réponse, échéance, date d'exécution au plus tôt, cadence, etc).
  - Fiabilité : le système répond à des contraintes de disponibilité.

$\Longrightarrow$ Système prédictible. Recherche des pires cas.

- **Un système temps réel n'est pas un système "qui va vite" mais un système qui satisfait à des contraintes temporelles.**

Présentation, définitions (2)

- Exemples de grandeur [DOR 91, DEM 99] :
  - La **milliseconde** pour les systèmes radar.
  - La **seconde** pour les systèmes de visualisation humaine.
  - Quelques **heures** pour le contrôle de production impliquant des réactions chimiques.
  - **24 heures** pour les prévisions météo.
  - Plusieurs **mois ou années** pour les systèmes de navigation de sonde spatiale.

TELECOM PARIS
Introduction aux systèmes temps réel

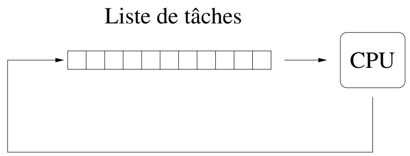
Présentation, définitions (3)

- Notion de systèmes temps réel **dur** et **souple**.
- **Systèmes embarqués** : systèmes informatiques temps réel n'impliquant pas l'intervention directe de l'homme (ex : la ligne 14 du Métro Parisien).
- Exemple d'applications :
  - Décodeurs numériques pour les réseaux TV.
  - SIG et systèmes de régulation automobile.
  - Terminal GSM.
  - Vidéo-conférence sur Internet.

Sous partie 1.2

Mécanismes et concepts de base

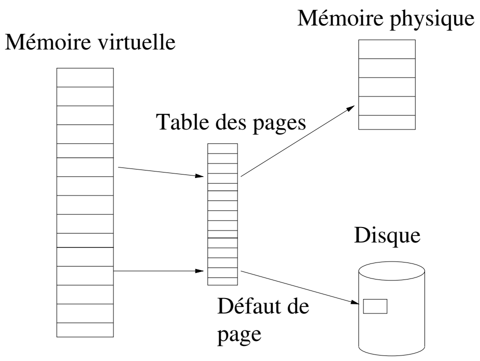
Vers le déterministe temporel (1)



- Ordonnancement (ex : Linux) :
  - Tâches bloquées, prêtes ou élues.
  - Priorité = nice + priorité dynamique proportionnelle au temps d'attente dans l'état "prêt".
  - Election haute priorité d'abord, round-robin (notion de quantum).
  - Equité, pas de prise en compte de l'urgence ou de contrainte temporelle. Politique généralement opaque. Temps de réponse inconnu.

$\Longrightarrow$ Utilisation d'ordonnanceur spécialisé (priorité, préemption, etc).

Vers le déterministe temporel (2)

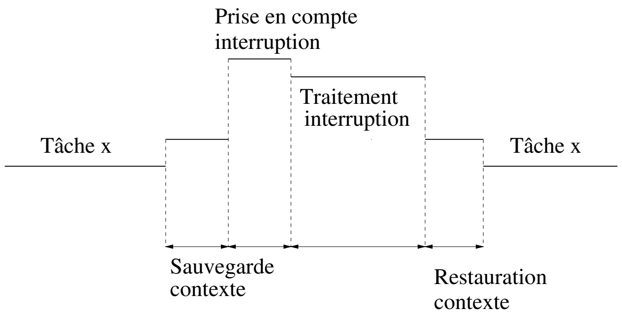


- Mémoire :
  - Nombre de défauts de page inconnu. Durée inconnue. Opération très coûteuse.
  - Impact sur l'ordonnancement.
  - Caches mémoire, caches disque. Allocation dynamique (*First Fit, Best Fit*). Idem pour disque.

$\Longrightarrow$ Modèle mémoire plat, allocation statique. Vérrouiller les pages en mémoire. Désactiver les caches.

Introduction aux systèmes temps réel

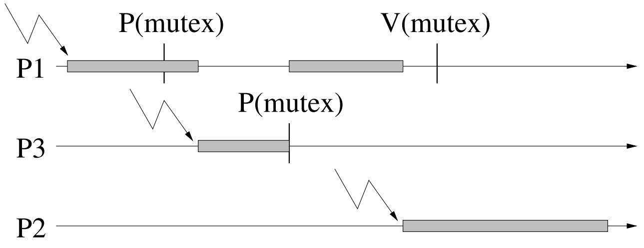
Vers le déterministe temporel (3)



- Interruptions et préemptivité :
  - Priorité des interruptions modifiable par le concepteur.
  - Nombre d'occurrences inconnu. Acquisition et traitement successifs $\Longrightarrow$ éventuellement long.
  - UNIX = verrouillage important, peu préemptif.

$\Longrightarrow$ Découplage acquisition et traitement des interruptions. Co-gestion tâches/interruptions. Noyau préemptif.

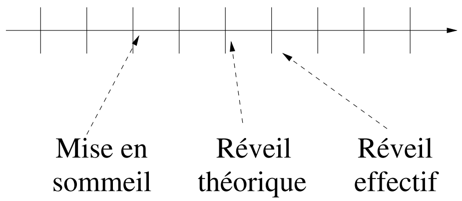
Vers le déterministe temporel (4)



- Partage de ressources :
  - Sémaphore = compteur + file d'attente.
  - Inversion de priorité : une tâche de faible priorité bloque une tâche de plus forte priorité. Temps de blocage inconnu.
  - Gestion FIFO (*First In, First Out*) $\Longrightarrow$ accès non prioritaire aux ressources.
  - Impact sur l'ordonnancement.

$\Longrightarrow$ Héritage de priorité (simple ou plafonné). Allocation ordonnée.

Vers le déterministe temporel (5)



- Gestion du temps, besoin de précision :
  - Mesure du temps et réveil des tâches $\Longrightarrow$ mesure = meilleur précision que réveil.
  - Précision dépend de l'architecture et du système d'exploitation (ex : Linux $\geq$ 2.2.3 + Pentium ; réveil $\geq 10\ ms$; mesure en μs).
  - Circuit d'horloge programmé pour générer des interruptions périodiques. Choix période.

$\Longrightarrow$ Programmation du circuit (ex : sans utiliser de période, ou par la biais de matériels spécialisés).

Sous partie 1.3

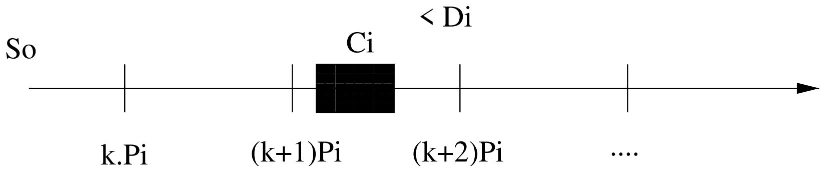
Ordonnancement et modèles de tâches

Ordonnancement, définitions (1)

- Objectifs : prendre en compte les besoins d'urgence, d'importance et de réactivité des applications temps réel.
- Critère de classement :
  - Hors ligne/en ligne. Moment où sont effectués les choix d'allocation.
  - Priorités statiques/dynamiques. Les priorités changent elles ?
  - Préemptif ou non. Tâche interruptible par d'autres ?
  - Oisif ou non. Système inactif ?
- Recherche des propriétés d'optimalité, d'ordonnançabilité.

Ordonnancement, définitions (2)

- Modèles de tâches : activations, tâches périodiques, tâches sporadiques, tâches apériodiques.



- Modèle de tâches de Lui et Layland [LIU 73] :
  - Tâches périodiques et indépendantes.
  - Borne sur le temps d'exécution : C_i (capacité).
  - Période d'activation : P_i.
  - Echéance sur requête : $D_i = P_i$.

Ordonnancement, définitions (3)

- Deux phases : calcul des priorités + élection.

1. Phase de calcul à partir :
   - D'une période : Rate monotonic (RM).
   - D'une échéance : Deadline monotonic (DM), Earliest Deadline First (EDF).
   - D'une laxité : Least Laxity First (LLF). Laxité = échéance - durée exécution restante.
2. Phase d'élection :
   - Plus haute priorité.
   - Round-robin.

L'algorithme Rate Monotonic (1)

– Priorités fixes. Priorités = inverse de la périodicité. Election de la plus forte priorité.

– Tâches périodiques uniquement.

– Analyse hors ligne $\Longrightarrow$ applications statiques.

– Cas préemptif $\Longrightarrow$ ordonnançabilité =

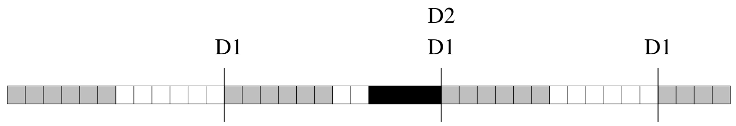
$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq n(2^{\frac{1}{n}} - 1)$$

– 100 % si tâches harmoniques (périodes multiples entre elles).

L'algorithme Rate Monotonic (2)

- Exemple (cf. [DEM 99]) :

T1 : C1=6 ; P1=12 (gris)
T2 : C2=8 ; P2=24 (blanc)
Noir = libre



L'algorithme EDF (1)

– Priorités dynamiques. Election de la plus courte échéance d'abord.

– Tâches périodiques et apériodiques + en ligne $\Longrightarrow$ mieux adapté que RM aux applications dynamiques.

– Cas préemptif $\Longrightarrow$ optimal, ordonnançabilité =

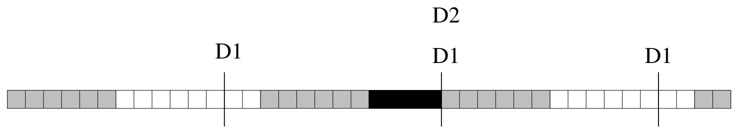
$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1$$

– Instable en sur-charge.

L'algorithme EDF (2)

- Exemple (cf. [DEM 99]) :

T1 : C1=6 ; P1=12 (gris)
T2 : C2=8 ; P2=24 (blanc)
Noir = libre



Extensions étudiées

- Algorithmes mixtes (partie statique et dynamique).
- Contrainte $D_i = P_i$. Ajout de contraintes d'exécution au plus tôt.
- Tâches dépendantes (contraintes de précédence, communications, accès successifs à des ressources).
- Partage de ressources (inversion de priorité et temps de blocage maximum).
- Prise en compte des tâches apériodiques (ex : technique du serveur sporadique [SPR 89]).

Sous partie 1.4

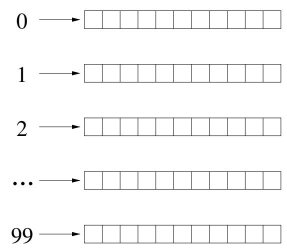
Mise en œuvre des systèmes temps réel

Mise en œuvre

1. Environnement d'exécution et de développement :
   - La norme POSIX.
   - L'approche moniteur.
   - Systèmes d'exploitation temps réel.
   - Services "temps réel" pour UNIX.
   - L'approche micro-noyau.
2. Langages, méthodes formelles.

La norme POSIX

- POSIX = standard ISO/ANSI définissant une interface portable de systèmes d'exploitation.
- POSIX.4 [GAL 95] = extensions temps réel.
- Ordonnancement, gestion du temps (temporisateur, mesure), synchronisation (sémaphores), E/S, IPC (pipe, file de messages, signaux, segment de mémoire).
- Ordonnancement :



  - Priorités fixes, préemptif $\Longrightarrow$ RM facile.
  - Une file d'attente par priorité + politiques de gestion de la file (*SCHED_FIFO*, *SCHED_RR*, *SCHED_OTHERS*).

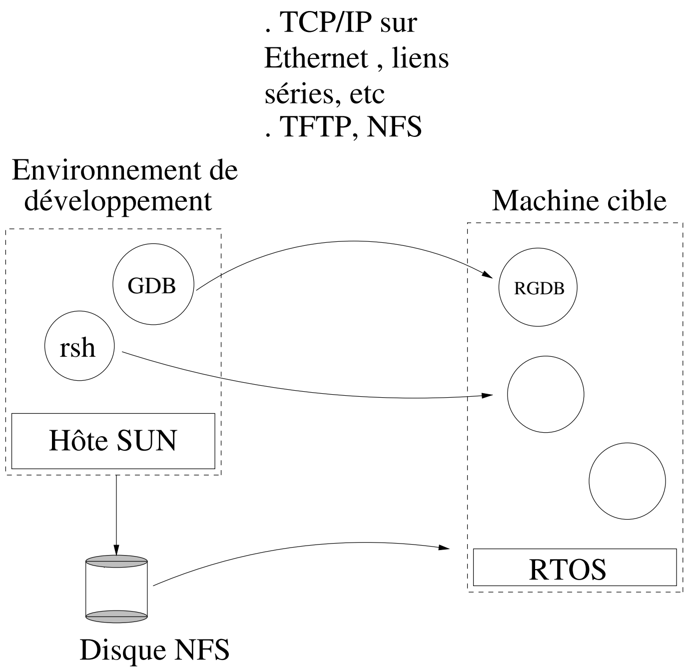
"Moniteur" et OS temps réel (1)

1. Moniteur :
   - Système de faible envergure (ex : pour application embarquée).
   - Collection de primitives $\Longrightarrow$ édition des liens.
   - Parfois mono-programmé (systèmes cycliques, calendrier). Avec ou sans protection mémoire.
   - Environnement de compilation croisée.
2. Système d'exploitation :
   - Interaction par appels systèmes.
   - Système multi-programmé (souvent POSIX).
   - Avec ou sans environnement de compilation croisée.

$\Longrightarrow$ La limite est aujourd'hui floue. Beaucoup de produits font les deux.

"Moniteur" et OS temps réel (2)

- La compilation croisée :



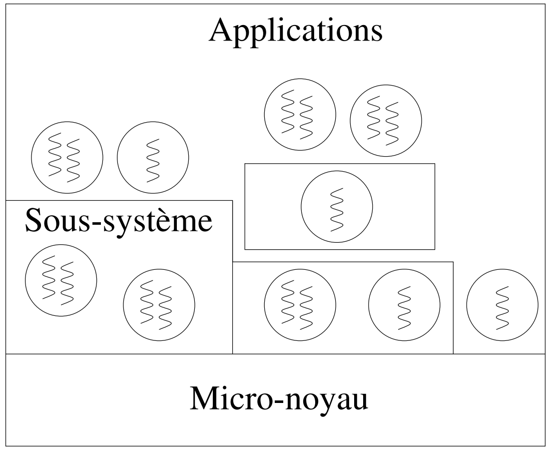
Services "temps réel" pour UNIX

- Nouvelles contraintes : déterminisme temporel plus faible (matériel et logiciel) et cohabitation avec des applications temps partagé.

$\Longrightarrow$ Applications peu fortement contraints.

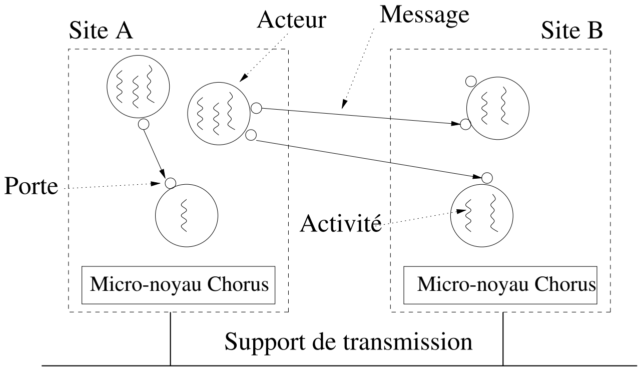
- Points faibles :
  - Préemptivité réduite (Linux) sauf conception adaptée (Solaris), ou points de préemption.
  - Inversion de priorité possible.
  - Présence de cache, de mémoire virtuelle $\Longrightarrow$ verrouillage possible des pages.
- Points forts :
  - Souvent à la POSIX (ordonnancement).
  - Environnement évolué (cf. hôte/cibles).
  - Intégration technologies non temps réel : IHM, BD.

L'approche micro-noyau : Chorus (1)



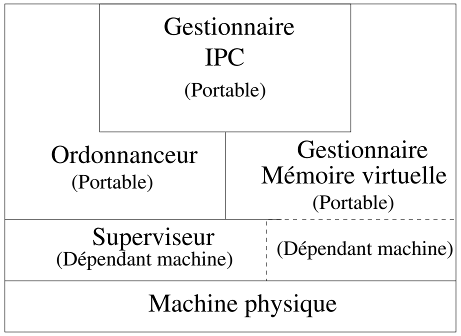
- Système monolithique : abstractions figées. Politiques de gestion de ressources figées et opaques. Empreinte mémoire importante.
- Micro-noyau : personnalités et compatibilité binaire. Modèle Client/serveur. Moindre performance mais plus grande flexibilité $\Longrightarrow$ prise en compte des besoins applicatifs. Empreinte mémoire minimale.

L'approche micro-noyau : Chorus (2)



• Abstractions : site, acteurs, threads, portes, messages, segments [ROZ 91].

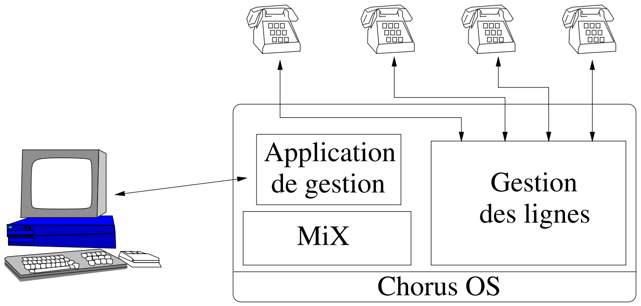
L'approche micro-noyau : Chorus (3)



• Personnalités : Jazz (JAVA), MiX (UNIX SVR4), COOL (CORBA 2), ClassiX (libC, TCP/IP, fichiers).

• Micro-noyau portable et modulaire dans les nouvelles versions. Ne comprenant que les mécanismes de base et non les politiques de gestion des ressources.

L'approche micro-noyau (4)



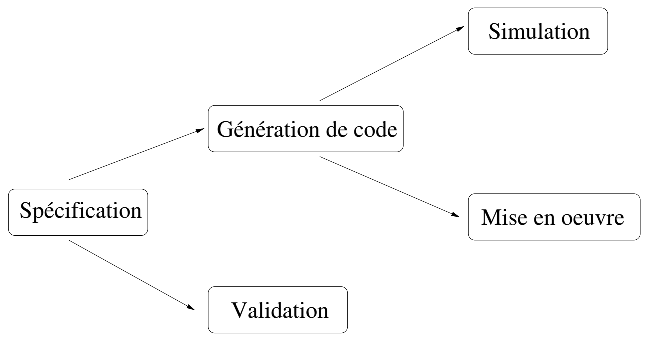
– Fort déterminisme temporel pour la partie télécommunication.

– Services de type "temps partagé" pour la partie gestion.

Outils de développement

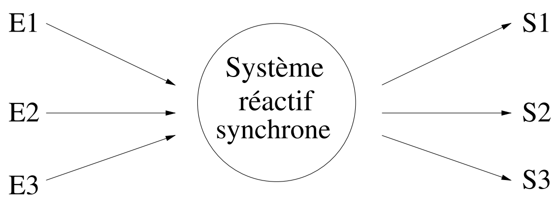
- Principaux critères de choix :
  - Déterminisme temporel et logique.
  - Présence d'abstractions "temps réel" (tâches, IPC).
  - Accès aux ressources de bas niveau.
  - Portabilité, normalisation.
- Exemples de solutions :
  - Langages de bas niveau : C, Assembleur. Utilisation directe des services du système. Accès direct aux ressources de bas niveau. Langage généralement restreint (sous-partie ayant un comportement temporel déterministe facile à évaluer).
  - Langages de haut niveau : ADA (abstractions temps réel + runtime ou lien avec le système sous-jacent).
  - Langages formels avec génération de code.

Les approches formelles



- Objectifs :
  - Spécifier formellement un système.
  - Valider des propriétés (ex : logique temporelle).
  - Génération de code pour effectuer des simulations.
  - Etre capable d'embarquer le logiciel (ex : problèmes de taille).

L'approche synchrone (1)



- Système réactif : fonctionnement cyclique. Atomicité. Déterministe logique.
- Système synchrone : temps d'exécution nul = temps de réaction plus court que le temps inter-arrivée des événements.
- Temps discrétisé = facilite sa manipulation.

L'approche synchrone (2)

- Exemple : le langage Esterel [BER 98].
  - Langage impératif.
  - Manipulation aisée du temps.
  - Notion de tâche mais parallélisme d'expression.
  - Notion de signal (externe, interne entre tâches).
  - Déterministe logique et temporel du code généré (automates, circuits).
  - Intégration et génération de code en C/ADA $\Longrightarrow$ Esterel = contrôle uniquement.
  - Simulation/validation.

L'approche synchrone (3)

```
module reveil :
   constant ATTENTE : integer;
   input SECONDE;
   input ANNULATION;
   output MINUTE;
   output ALARME;

   every 60 SECONDE do
        emit MINUTE;
   end every
   ||
   loop
        do  await ATTENTE SECONDE;
              emit ALARME;
        watching ANNULATION;
   end loop
end module
```

Partie 2

Exemple : les applications multimédias réparties

Sommaire

Sous partie 2.1

Présentation, définitions

Quelques définitions

1. Médium = moyen par lequel l'information est perçue, stockée ou transmise (définition du MHEG[a] de l'ISO).
2. Multimédia = littéralement qui est constitué de plusieurs média (pluriel de médium).
3. Médium de représentation, de stockage, de transmission, etc.

• Application multimédia : qui manipule plusieurs média de représentation.

[a] Multimedia and Hypermedia Expert Group.

La qualité de Service

- Notion de Qualité de Service (QoS, *Quality of Service*) : décrit la qualité d'un service donné. Garantie ou non (déterministe ou probabiliste).
- Ex : restitution d'un médium de représentation $\Longrightarrow$ QoS dit "utilisateur".

1. Contraintes de fiabilité : taux d'erreurs, taux de pertes, probabilité de disponibilité.
2. Contraintes spatiales : débit, taille, volume. Variable ou fixe.

Contraintes "temps réel" :

3. Contraintes temporelles : gigue, délais de bout en bout, temps de réponse. Synchronisations : intra et inter-flux, entre objets, etc.

Exemples de contraintes, besoins

- Contraintes spatiales (débit fixe ou variable) :
  - Audio qualité voie téléphonique : 64 Kbits/s.
  - Audio qualité CD : 1,4 Mbits/s.
  - Vidéo MPEG-1 : environ 1,5 Mbits/s, débit supérieur si MPEG 2.
  - Vidéo TVHD non compressé : 1,2 Gbits/s.
- Contraintes temporelles :
  - Intra-flux vidéo : cadence 40 ms ou 33 ms, gigue $\leq$ 120 ms.
  - Voix-lèvres : gigue $\leq +/- 80$ ms.
  - Téléphonie : latence $\leq$ 250 ms.

Exemples d'applications multimédias

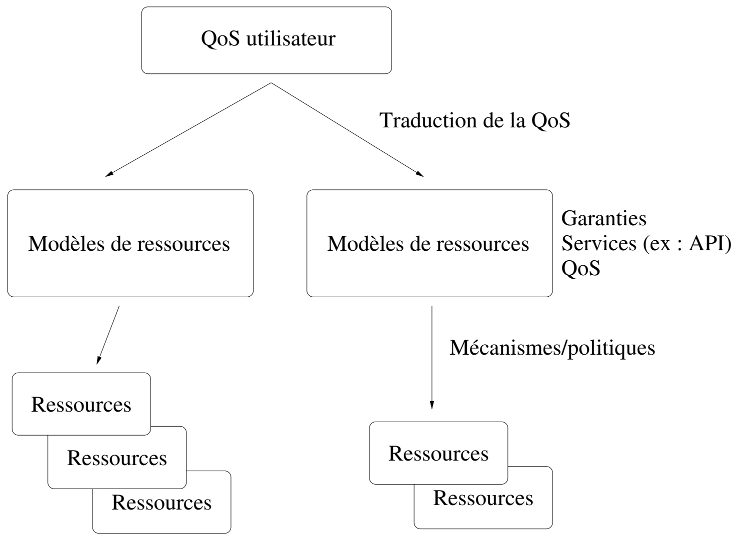
- Grande diversité : du temps réel fortement contraint aux applications partiellement déterministes.
- Dépend des : données, usages, environnements.
  - Exemple 1 : vidéo-conférence. Plate-forme dédiée. Opération chirurgicale $\Longrightarrow$ fort déterminisme. H.261. Débit fixe. RNIS.
  - Exemple 2 : télévision interactive.
    - Plate-forme généraliste. Non déterminisme temporel à cause du matériel et du logiciel (ex : PC + windows).
    - Application interactive.
    - Nombre de flots inconnu et QoS modifiable.
    - Débits variables et difficiles à estimer hors ligne.

Système d'exploitation multimédia

- Objectif : gestion des ressources et prise en compte des contraintes de QoS. Tâche complexe car :
  - Nécessite de nouvelles abstractions (ex : flots continus).
  - Ressources nombreuses et services de QoS très différents. QoS de bout en bout.
  - Cohabitation avec les applications temps partagé.
- Et dans l'idéal :
  - Masquer la complexité au concepteur $\Longrightarrow$ sinon nécessite compétences étendues.
  - Fonctions d'intégration (logiciels).
  - Portable, maintenable.

$\Longrightarrow$ Utilisation d'un modèle global.

Des besoins aux services (1)



Des besoins aux services (2)

- Le système doit donc offrir les services suivants :

1. Expression des contraintes de QoS, des besoins. Traduction automatique.
2. Réservation des ressources, contrôle d'admission.
3. Allocation des ressources.
4. Supervision : estimation, rétroaction et pilotage, "policing".

- Implantation "répartie" : matériel, système d'exploitation. Intégration de services $\Longrightarrow$ espace utilisateur, middleware.

Sous partie 2.2

Expression de la QoS

Expression de la QoS

- Expression de la QoS utilisateur et de la QoS des éléments du système.
- Formalimes et puissance d'expression dépendent de l'élément concerné. Compromis nécessaires pour la traduction.
- Exemples :

1. API (ex : couche transport).
2. Structure syntaxique (FLEX [LIN 88], SMART [NIE 97]).
3. Méthodes formelles : possibilité de preuves (Réseaux de Petri, logiques, etc). Nouveau compromis. Principalement deux approches : événementielle et intervalle.

Le modèle du CNET/Lancaster (1)

- Expression de contraintes déterministes et probabilistes avec la logique temporelle QL [BLA 98a].
- Flux de données continues, événements, horloges.
- Exemples de contraintes déterministes :

1. Synchronisation inter-flux. Dérive (rafales non bornées).

$$\forall\, n,k \;:\; x \leq \tau(e, n+k) - \tau(e', n) \leq y$$

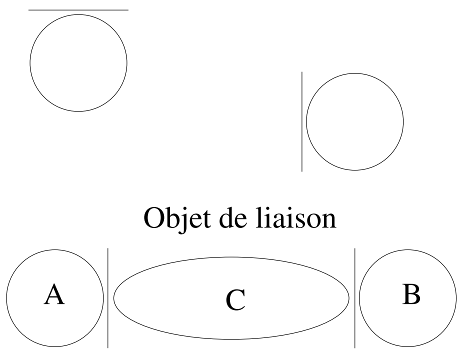
2. Synchronisation intra-flux. Ne dérive pas.

$$\forall\, n \;:\; x \leq \tau(e, n) - \tau(H_p, n) \leq y$$

3. Flux de données "strictement périodique".

$$\forall\, n \;:\; \tau(e, n) = n * p$$

Le modèle du CNET/Lancaster (2)

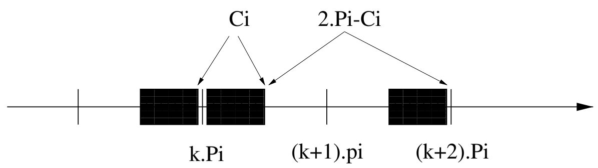


- Objets, objets de liaison [BLA 98a].
- Contrat de QoS, QoS requise et QoS offerte.

Sous partie 2.3

Gestion des ressources

Ordonnancement : modèles de tâches



• Le modèle périodique est il adapté aux applications multimédias ? Gigue $= 2.P_i$.

1. Contrainte de distance [SAK 95, HAN 96] :

$$\epsilon_1 \leq \tau(i, n+1) - \tau(i, n) \leq \epsilon_2$$

   Dimensionnement de la période, algorithme oisif, ajout d'une contrainte artificielle.

$$P_i = \frac{\epsilon_2 + C_i}{2}$$

2. Cadence d'activation [JEF 95] : plusieurs activations dans une période. Echéances relatives. Adaptation du nombre d'activations en fonction des ressources.

Ordonnancement : dynamicité

• Arrivée de nouvelles tâches, modification de la QoS, non conformité aux spécifications (systèmes sans "policing"). Exemples de mécanismes :

- Technique du serveur sporadique [SPR 89].
- Basculement "passif". Critère d'importance. Niveaux de QoS. Avec ou sans réservation (ex : Pegasus [SIJ 96], Brandt et al. [BRA 98]).
- Renégociation : modification des besoins de l'application (YARTOS [JEF 92]).
- Supervision + adaptation. Avec réservation préalable (Jones et al. [JON 97]) ou sans. Application adaptative.

Les ressources réseaux

• Définition : réseaux isochrores, synchrones et asychrones.

• Deux approches selon la QoS offerte :

1. Réservation de ressources pour les réseaux avec garantie de service (bande passante, comportement temporel). Orientés connexions.
2. Supervision et rétroaction pour les réseaux "best effort" (non déterministe).

• Mécanismes présents de l'application en passant par la couche transport, la couche liaison, etc...

Ex. de LAN déterministes (1)

- Solution principale aujourd'hui : ATM
  - Maturité naissante : portabilité, support parfois partiel (paramètres, services).
  - Essentiellement services CBR et VBR-rt :
    - CBR : (Constant Bit Rate). Débit et comportement temporel garantis. Trafic et QoS (CBR, PCR, maxCTD, ppCDV, CLR).
    - Gigue d'une grande précision $\Longrightarrow$ synchronisation.
    - VBR rarement offert (débit moyen + multiplexage statistique).
  - Prise en compte de la QoS soit en ATM natif (AAL5), soit IP "à la sauce QoS", éventuellement avec Classical-IP.

Ex. de LAN déterministes (2)

1. L'outsider : FDDI.
   - Trafics synchrone et asynchrone. Traduction de la QoS simple. Partage de la bande synchrone.
   - Borne déterministe (2*TTRT).
   - Compromis efficacité/valeur du TTRT. TTRT grand = gigue grande.
   - Backbone, donc rarement de bout en bout.
   - En perte de vitesse.
2. Challenger : le bus 1394.
   - Connexion de périphériques variés et "grand public".
   - Haut débit (jusqu'à 400 Mbits/s).
   - Services isochrones et asynchrones $\Longrightarrow$ logiciels pour les services isochrones rarement disponibles aujourd'hui.

Réseaux non déterministes (1)

- Les protocoles RTP/RTCP [SCH 96] :

1. RTP (Real Time Protocol) : Protocole utilisateur. N'offre pas de garantie de QoS. Utilisation de couche transport à faible coût (ex : UDP). Unicast et multicast. Données comprenant des contraintes temporelles $\Longrightarrow$ estampille horloge logique ou horloge globale (synchronisation, prise en compte de la gigue).
2. RTCP (Real Time Control Protocol) : Transfert de données de supervision (es : perte de paquet (congestion), estimation de délais de bout en bout, etc).

- RFC mais philosophie application ALF (Application Level Framing).

TELECOM PARIS
Introduction aux systèmes temps réel

Réseaux non déterministes (2)

1. L'approche intserv [SHE 97, WRO 97] :
   - Réservation (protocole RSVP). Contrôle d'admission.
   - Services temps réel, services élastiques.
   - Mise a l'échelle difficile : état/routeur + mécanisme de bout en bout ($\Longrightarrow$ LAN ?).
2. L'approche diffserv [BLA 98b] : différenciation de services.
   - Classification et agrégation de trafics. Champ DS dans le paquet IP.
   - Pas (ou peu) de signalisation.
   - Mise a l'échelle plus aisée.
   - Notion de domaine.

La gestion des supports de masse (1)

- Principalement magnétique/optique [DEM 99].
- Caractéristiques :
  - Progrès significatif en terme de volume.
  - Débit toujours insuffisant.
- Critères de performance :
  - Délai de rotation (fixe pour le magnétique, variable pour l'optique).
  - Temps de transfert (lecture/écriture).
  - Temps d'accès (10 ms pour le magnétique, 200 ms pour l'optique) $\Longrightarrow$ paramètre principalement ciblé.

La gestion des supports de masse (2)

- Problématique :

1. Débit ; systèmes de disques parallèles (RAID [PAT 88]).
2. Déterminisme temporel : minimisation de la gigue ou de la latence.
   - Principal indéterminisme : le temps d'accès.
   - Ordonnancement des requêtes (EDF, SCAN, SSTF).
   - Placement des données (contigues pour les flots continus) $\Longrightarrow$ fragmentation interne.

La gestion de la mémoire

- Parent pauvre dans les systèmes multimédias. Deux problèmes traités :

1. Dimensionnement : dans les extrémités : compensation de gigue (t=2*gigue*débit), partage d'informations et traitement (pipe-line, taille fonction du format de données traité plutôt que de la QoS).

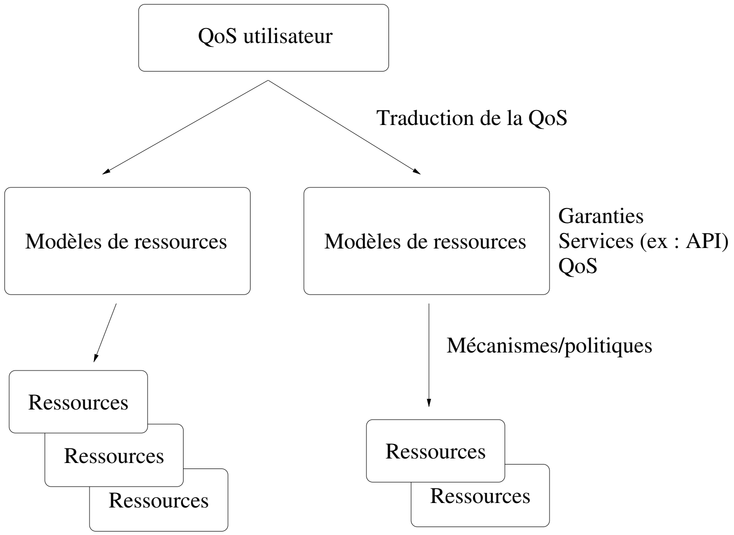
2. Allocation. Actuellement statique. Peu de solutions pour allocation dynamique et ramasse-miettes (ex : avec RM [HEN 97]). Problèmes principaux $\Longrightarrow$ fragmentation, concurrence avec le ramasse-miettes.

   $\Longrightarrow$ et pourtant, applications dynamiques, Java temps réel.

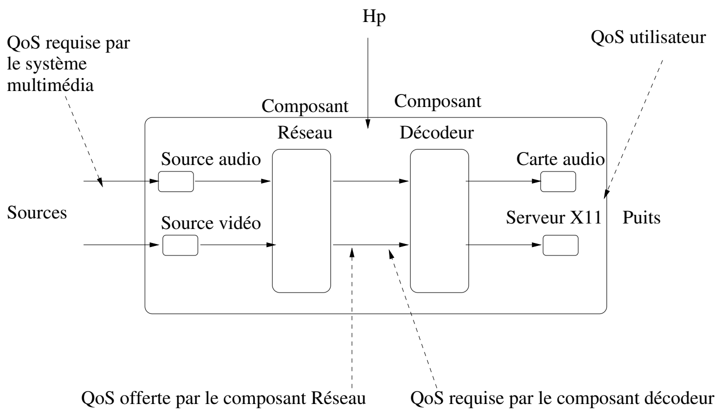
Sous partie 2.4

Traduction de la QoS

Les techniques de traduction de QoS



Exemple : CNET/Lancaster (1)



- Exemple : l'application de vidéo-conférence.

Exemple : CNET/Lancaster (2)

- Exemple : horloge H avec une période de 40 ms.

$$-10\ ms \leq \tau(i,n) - \tau(H,n) \leq 10\ ms$$

1. Ressource processeur :
   - Période (40 ms).
   - Echéance et date d'exécution au plus tôt (par rapport à la période).
2. Ressource réseau (ATM/CBR) :
   - PCR (1/40 * Taille i).
   - ppCDV (40 ms).
3. Mémoire : calculer en fonction de la gigue.

Partie 3

Conclusion, synthèse et perspectives

Conclusion, synthèse et perspectives

- Synthèse :

1. Système temps réel : systèmes contraints par le temps.
   - Mécanismes, ordonnancement.
   - Mise en œuvre.
2. Système multimédia : système manipulant plusieurs médias différents $\Longrightarrow$ besoins spécifiques vis-à-vis des applications temps réel :
   - Expression et traduction de la QoS.
   - Autres ressources impliquées.
   - Cohabitation applications temps partagé.
   - Systèmes dont le déterministe temporel est partiel.

Conclusion, synthèse et perspectives

- Bilan et perspectives :
  - Temps réel fortement contraint : bien maîtrisé. Modèle formel et validation. Répartition.
  - Multimédia : $\Longrightarrow$ systèmes temps réel ouverts
    - Nombreux mécanismes et politiques proposés. Applications de + en + dynamiques.
    - Manque d'intégration $\Longrightarrow$ complexité = frein à la diffusion de la technologie.
    - Comportements stochastiques $\Longrightarrow$ problèmes non résolus, méfiance des utilisateurs.
    - Tendance middleware, UNIX temps réel.
  - Applications nouvelles : voix sur IP, environnements virtuels, applications mobiles (télécoms).

Partie 4

Références

[BER 98] G. Berry. « The Foundations of Esterel ». in Proof, Language and Interaction: Essays in Honour of Robin Milner, G. Plotkin, C. Stirling and M. Tofte, editors, MIT Press, 1998.

[BLA 98a] G. Blair et J. B. Stefani. *Open Distributed Processing and Multimedia.* Addison-Wesley, 1998.

[BLA 98b] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, et W. Weiss. « RFC2475 : An architecture for Differentiated Services ». Network Working Group, pages 1-33, December 1998.

[BRA 98] S. Brandt, G. Nutt, T. Berk, et M. Humphrey. « Soft Real-Time Application Execution with Dynamic Quality of Service Assurance ». International Workshop on Quality of Service (IWQOS'98), Napa - CA, May 1998.

[DEM 99] I. Demeure et C. Bonnet. *Introduction aux systèmes temps réel.* Collection pédagogique de télécommunications, Hermes, September 1999.

[DOR 91] A. Dorseuil et P. Pillot. *Le temps réel en millieu industriel.* Edition DUNOD, Collection Informatique Industrielle, 1991.

[GAL 95] B. O Gallmeister. *POSIX 4 : Programming for the Real World* . O'Reilly and Associates, January 1995.

[HAN 96] C. C. Han, K. J. Lin, et C. J. Hou. « Distance-constrained Scheduling and Its Applications to Real-time Systems ». *IEEE Transactions on computers*, 45(7):814–826, Jully 1996.

[HEN 97] R. Henriksson. « Predictable Automatic Memory Management for Embedded Systems ».

Dans Peter DICKMAN et Paul R. WILSON, éditeurs, *OOPSLA '97 Workshop on Garbage Collection and Memory Management*, October 1997.

[JEF 92] K. JEFFAY, D. L. STONE, et F. D. SMITH. «Kernel Support for Live Digital Audio and Video». *Computer Communications*, 15(6):388–395, August 1992.

[JEF 95] K. JEFFAY et D. BENNETT. « A Rate-Based Execution Abstraction For Multimedia Computing». *In Lectures Notes in Computing Science, T. D. C. Little and R. Gusella, Springer-Verlag, Heidelberg*, 1018:64–75, April 1995.

[JON 97] M. JONES, D. ROSU, et M. ROSU. « CPU Reservations and Time Constraints: Efficient, Predictable Scheduling of Independent Activities ». 16th ACM Symposium on Operating Systems Principles in Saint-Malo (SOSP'97) - France, October 1997.

[LIN 88] K. J. LIN et S. NATARAJAN. « Expressing and Maintaining Timing Constraints in Flex». pages 96–105. In Real Time Systems Symposium, December 1988.

[LIU 73] C. L. LIU et J. W. LAYLAND. «Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environnment». *Journal of the Association for Computing Machinery*, 20(1):46–61, January 1973.

[NIE 97] J. NIEH et M. LAM. « The Design, Implementation and Evaluation of SMART: A Scheduler for Multimedia Applications ». 16th ACM Symposium on Operating Systems Principles in Saint-Malo (SOSP'97) - France, October 1997.

[PAT 88] D. A. PATTERSON, G. GIBSON, et R. H. KATZ. A Case for Redundant Arrays of Inexpensive Disks (RAID). pages 109–116. June 1988.

[ROZ 91] M. ROZIER, V. ABRASSIMOV, et F. ARMAND. « Overview of the CHORUS Distributed Operating

Systems ». Rapport Technique, Chorus Systèmes, CS/TR-90-25.1, February 1991.

[SAK 95] M. SAKSENA, R. GERBER, et W. PUGH. «Parametric Dispatching of Hard Real-time Tasks». *IEEE Trans. on Computers*, 44(3):471–479, 1995.

[SCH 96] H. SCHULZRINNE, S. CASNER, R. FREDERICK, et V. JACOBSON. « RFC1889: RTP: A Transport Protocol for Real-Time Applications ». Network Working Group, pages 1-75, January 1996.

[SHE 97] S. SHENKER, C. PARTRIDGE, et R. GUERIN. « RFC2212: Specification of Guarantedd Quality of Service». Network Working Group, pages 1-19, September 1997.

[SIJ 96] P. SIJBEN et S. J. MULLENDER. «Quality of Service in Distributed Multimedia Systems». in Trends in distributed systems, Springer Lectures Notes on computing systems, TREVS 1161, 1996.

[SPR 89] B. SPRUNT, L. SHA, et J.P. LEHOCZKY. «Aperiodic task scheduling for hard-real-time systems ». *The Journal of Real Time Systems*, 1:27–60, 1989.

[STA 88] John STANKOVIC. «Misconceptions about real-time computing». *IEEE Computer*, October 1988.

[WRO 97] J. WROCLAWSKI. « RFC2210: The Use of RSVP with IETF Integrated Services ». Network Working Group, pages 1-31, September 1997.