Simulation d'un senseur radio-fréquences avec Xenomai

David Chabal - CS Toulouse

RMLL2008 – Atelier « embarqué »
Mont de Marsan - 2 juillet 2008

Sommaire

- Introduction
- Présentation du Senseur RF à simuler
- Description du besoin
- Pourquoi Xenomai ?
- Présentation de Xenomai
- Conception et développement du simulateur
- Aspects juridiques
- Bilan d'utilisation, autres applications

Introduction

- Cette présentation est un retour d'expérience basé sur deux projets industriels autour de Xenomai, une solution temps-réel dur adossée à Linux.

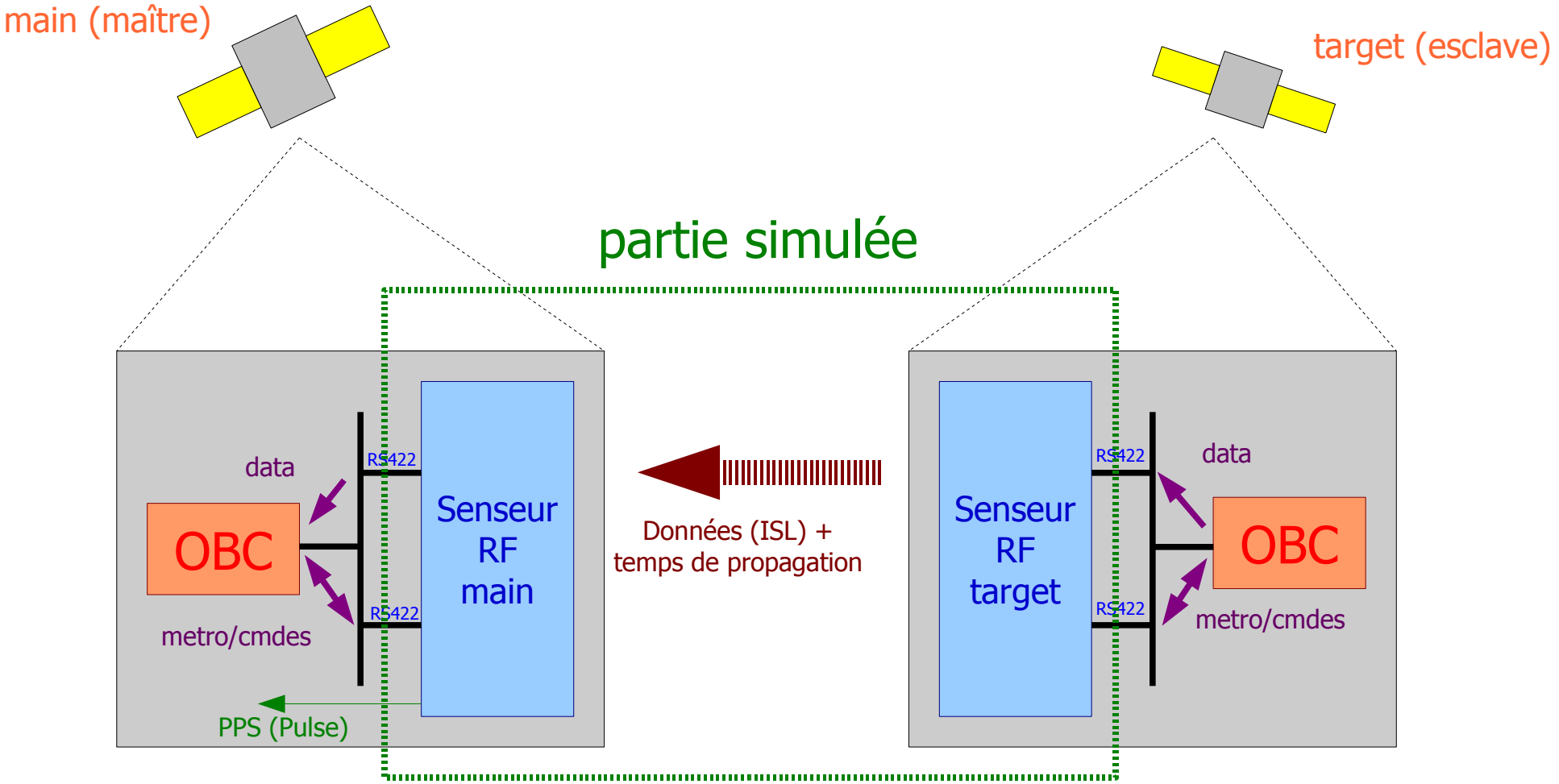
- Le domaine d'application est la simulation d'équipements embarqués sur des satellites.
- Pourquoi l'atelier « embarqué » des RMLL ? Deux raisons :
  - Souvent « temps-réel = embarqué »
  - « Logiciel (embarqué) temps-réel » = « Moyens de test temps-réel »
- Ces réalisations sont :
  - Un simulateur GPS dédié à la validation d'un logiciel de vol micro-satellite
  - Un simulateur de senseur radio-fréquences
  - Ce sont des applications de type « commande/contrôle ».

Présentation du Senseur RF à simuler

- L'agence spatiale suédoise a lancé la mission de démonstration technologique Prisma qui consiste en deux satellites volant en formation.
- La trajectoire d'un des deux satellites peut être asservie par l'autre.
- Le calcul de la position et le dialogue entre les deux satellites sont assurés par un senseur RF.
- Le CNES (Toulouse) contribue à cette mission en développant les algorithmes guidage, navigation et contrôle (GNC) basé sur un nouveau senseur RF conçu par Thalès Alenia Space.
- Pour valider ces algorithmes, il faut simuler l'environnement des satellites :
  - Interfaces HW/SW, comportement temporel,
  - Mécanique du vol : l'évolution de leur configuration géométrique (position et orientation relative)
- La simulation se fait en « boucle fermée ».
- Cette simulation utilise un banc de test réalisé par CS Toulouse.

Présentation du Senseur RF à simuler

- Les interfaces du senseur RF

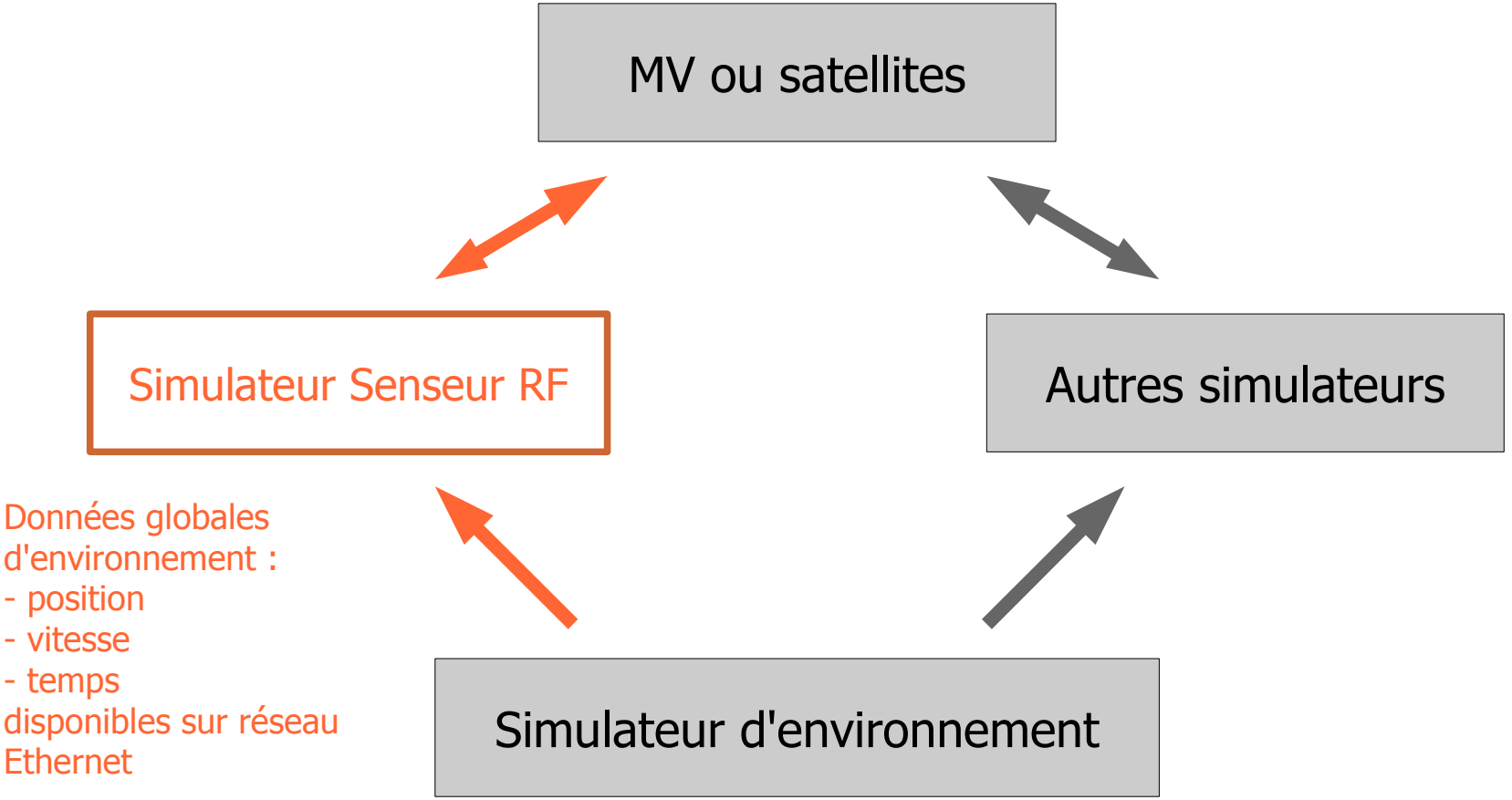


OBC: OnBoard Computer, calculateur de bord

Présentation du Senseur RF à simuler

Présentation du Senseur RF à simuler

- Intégration dans l'environnement de test



Description du besoin

- Contraintes temporelles
  - La plus critique : génération d'un signal PPS, pulse TTL à 1Hz, avec une précision de l'ordre de 1 ms
  - Le dialogue série (4 ports) qui doit être représentatif (pas de latence pathologique)
  - Fort déterminisme au vu de la lourdeur que représentent des tests sur satellite, contraintes temps-réel dur.
- Contraintes techniques
  - Intégration dans l'environnement de test du satellite (connectivité)
- Contraintes financières
  - Coût élevé des solutions commerciales (licences et matériels spécifiques)
- **Solution retenue**
  - PC équipé de cartes série RS422 du commerce
  - Linux : distribution Mandriva Free

Pourquoi Xenomai ?

- ... à la place de Linux tout seul ?
- Il existe plusieurs pistes pour faire du temps-réel avec Linux :
  - Les solutions « natives »
    - Configuration de la politique de préemption du noyau. Ajout de points de préemption à l'intérieur du noyau. Réduction de la latence moyenne, temps-réel mou.
  - La solution « patch »
    - Patch PREEMPT_RT d'Ingo Molnar : tout le noyau est préemptible
  - Les solutions « co-noyau »
    - Les contraintes RT ne sont plus gérées par Linux mais par un noyau dédié (co-noyau). Temps-réel dur.
- Les avantages de Xenomai
  - Maturité : plusieurs années d'existence (documentation de l'API...)
  - Logique industrielle : développement orienté vers la maintenabilité et la stabilité
  - Support très réactif (mailing list)

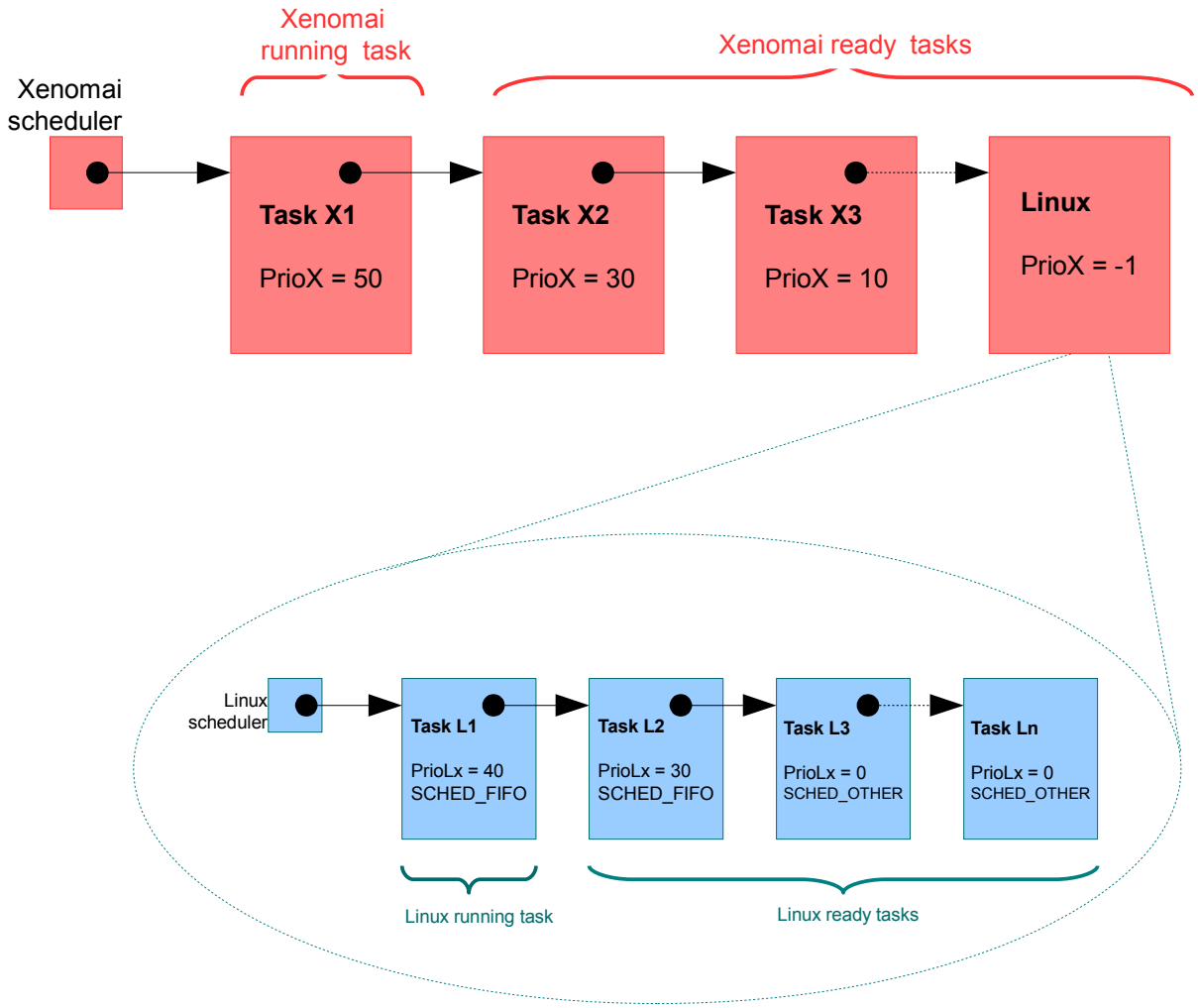
Présentation de Xenomai

- Le principe du co-noyau
  - Ajout d'un scheduler supplémentaire
  - L'installation de Xenomai passe par le patch des sources du noyau (licence).
  - Les interruptions sont prises en charge par une couche appelée ADEOS (hyperviseur).
  - Objets spécifiques Xenomai : tâches, sémaphores, files de messages...(≠IPC) d'où une API et des librairies

- Les règles du scheduler Xenomai
  - Prend en charge les tâches RT (sous certaines conditions)
  - Exécute Linux en tâche de fond

Présentation de Xenomai

- Exemple de co-scheduling (les priorités sont croissantes) :

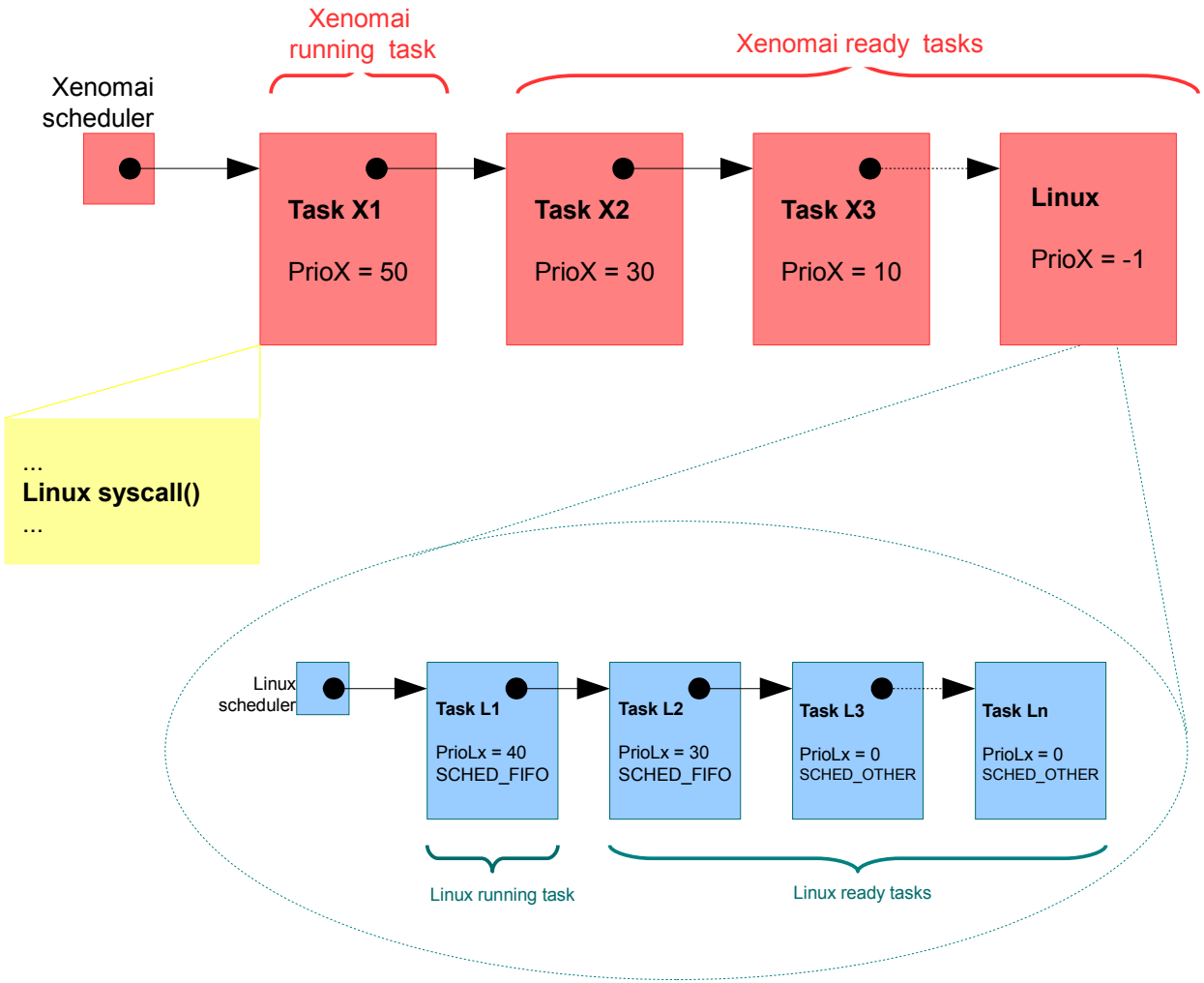


Présentation de Xenomai

- La coexistence des deux OS
  - Comment traiter les appels systèmes Linux depuis les tâches Xenomai ?
    Exemple : récupération de la date courante.
  - Solution : déplacer les tâches Xenomai entre les ordonnanceurs en fonction des ressources accédées.
  - Un peu de terminologie :
    - Tâche Xenomai dans l'ordonnanceur Xenomai : **primary mode**
    - Tâche Xenomai dans l'ordonnanceur Linux : **secondary mode**
  - Règles fondamentales :
    - **1. Une tâche Xenomai conserve sa priorité, quelque soit le scheduler dans laquelle elle se trouve.**
    - **2. Dans le scheduler Linux, la conservation des priorités utilise la classe SCHED_FIFO avec une correspondance directe.**

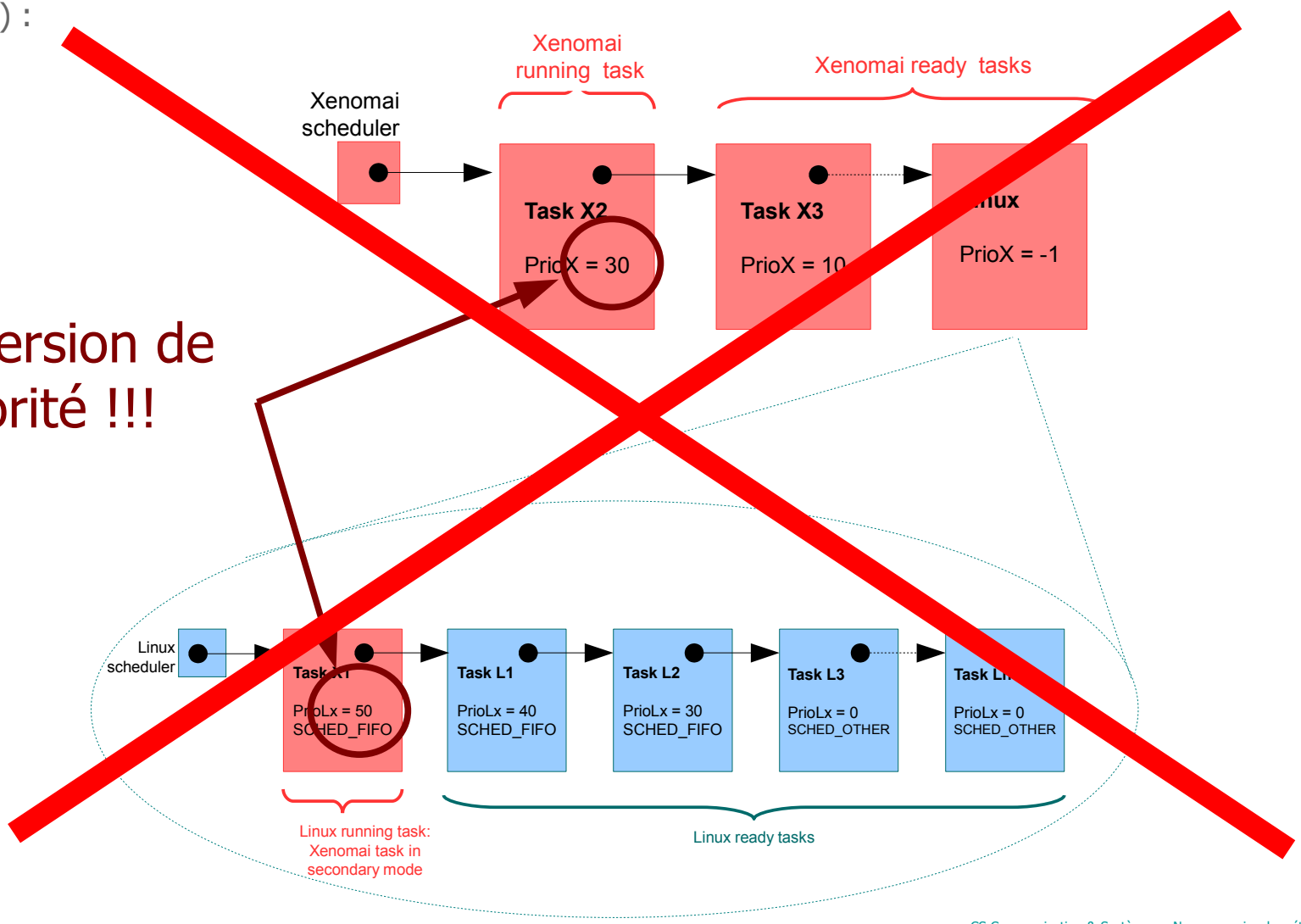
Présentation de Xenomai

- Exemple de scheduling, X1 en primary mode et qui fait un appel système Linux :



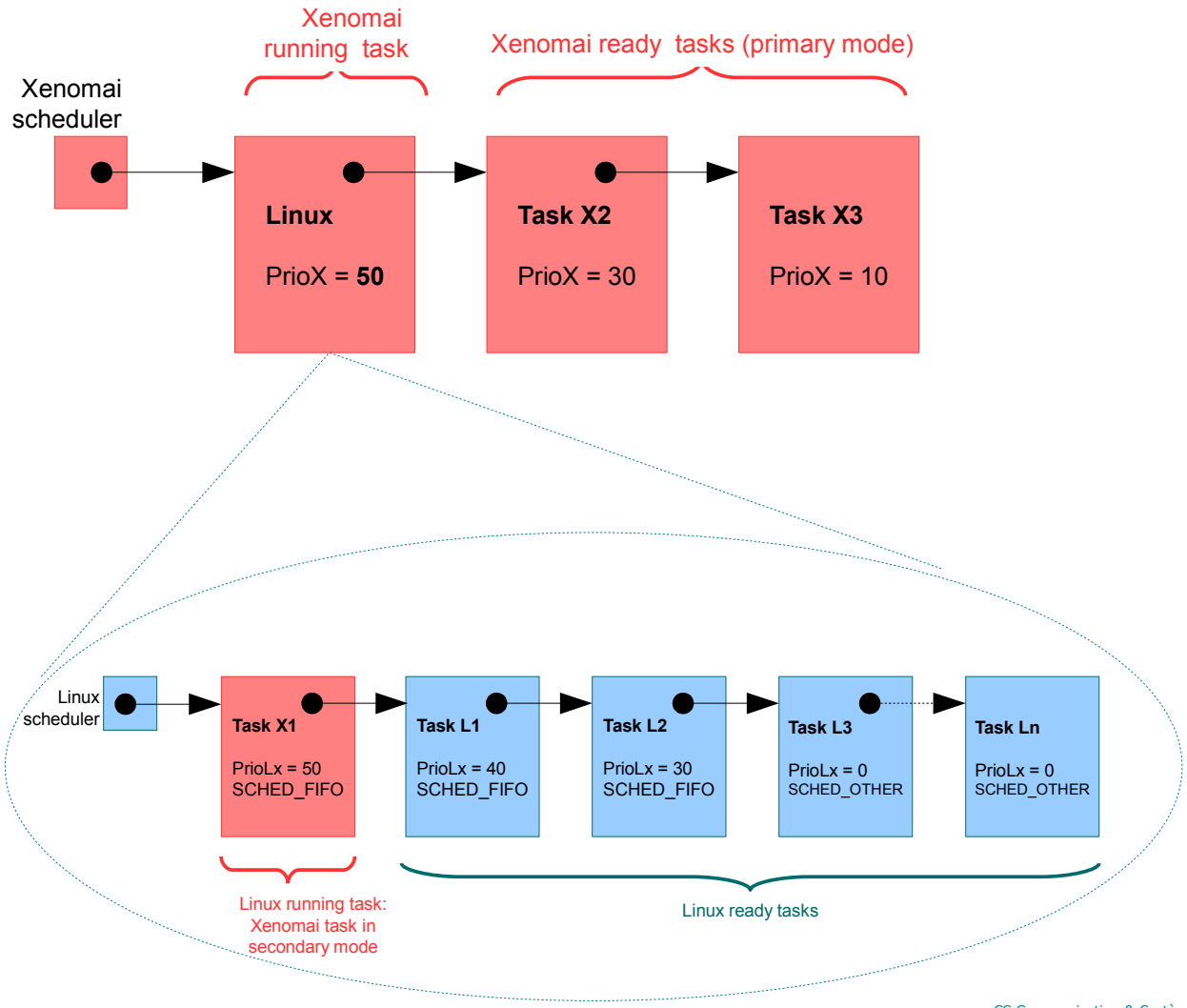
Présentation de Xenomai

- Contre-exemple de scheduling, tâche X1 en secondary mode (ce qu'il se passerait si la priorité n'était pas conservée) :



Présentation de Xenomai

- Exemple de scheduling, tâche X1 en secondary mode (**les priorités sont conservées**) :



Présentation de Xenomai

- Problème
  - En secondary mode, on retombe sur les travers de Linux pour finalement faire du temps-réel mou:
    - Préemption possible par des tâches de plus haute priorité
    - Préemption par les IT (top-half) et les tasklets (bottom-half)
    - Latence intrinsèque du noyau (manque de points de descheduling)
- Solution
  - Les tâches « critiques » doivent impérativement rester en primary mode.
  - **Dans la phase de conception, il faut identifier les ressources Linux accédées afin de connaître quel sera le mode des tâches.**

    Et ce n'est pas toujours évident car il faut traquer les appels système Linux !!!
  - Les tâches en secondary mode doivent avoir une priorité :
    - Plus faible que les tâches en primary mode (car conservation des priorités à travers les schedulers).
    - Plus forte que les tâches Linux avec lesquelles elles sont en concurrence (classe SCHED_FIFO)

Conception et développement du simulateur

- Méthodes et outils
  - Cycle en V classique
  - Conception textuelle et Simulink
  - Une grande partie du code (C) est générée par RTW à partir du modèle MATLAB/Simulink.
- Conception de l'architecture dynamique : spécificité
  - Identification des tâches Xenomai utilisant des ressources Linux, mise en place de mécanismes de communication *ad hoc* (risque de basculement).
- Avantages
  - Environnement rapidement opérationnel :
    - gcc présent dans la distribution Linux : pas de compilation croisée, pas de problème de licence...
    - Connectivité réseau, FTP... immédiate
    - Gestion de configuration
  - Phase de conception : maquettage de l'architecture dynamique au plus tôt, validation de cette architecture
- Et inconvénients...

Conception et développement du simulateur

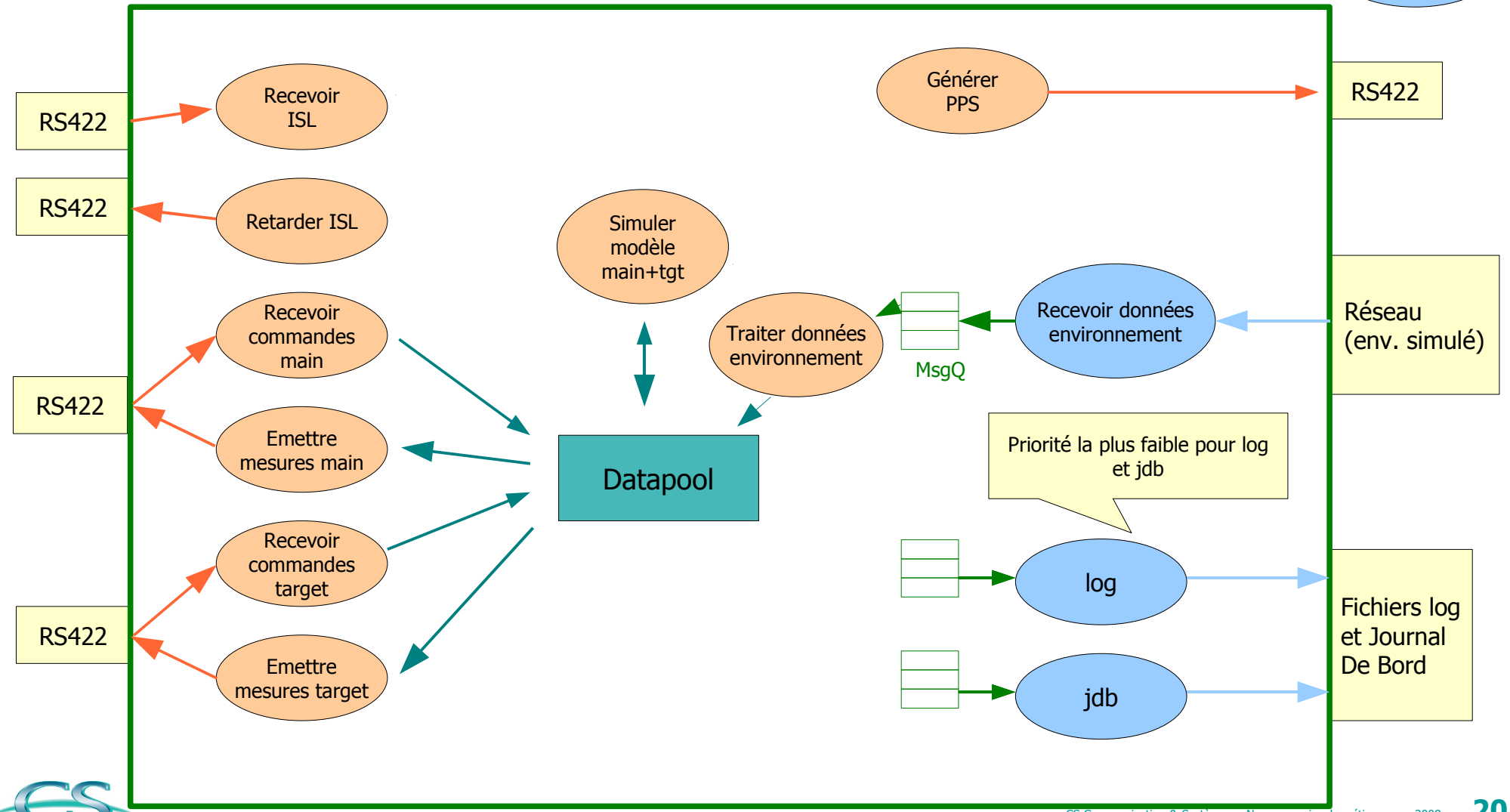
- Sur PC : « Intégration HW/SW » ou la configuration du noyau
  - Certaines options du noyau Linux peuvent interférer avec Xenomai en augmentant les temps de latence de manière pathologique.
  - Généralement, cela est dû à une mauvaise configuration du matériel par Linux (ou le BIOS), d'où la nécessité d'une configuration fine du noyau :
    - Configuration au plus proche du matériel (type de CPU, SMP)
    - Suppression des options d'économie d'énergie (frequency scaling)
    - Interruptions non masquables « SMI » (gestion d'énergie)
  - « Qualification » de la configuration grâce à un utilitaire présent dans la distribution Xenomai.
  - Architectures PC complexes : multitude de configurations matérielles (chips, fabricants...), approvisionnement de machines identiques ?

Conception et développement du simulateur

- Intégration du matériel « projet »
- Problème :
  - Accéder au matériel tout en restant en primary mode
  - Or, impossible d'utiliser les drivers Linux !!! (puisque appel système et basculement en secondary mode)
- Solution :
  - Développement de drivers spécifiques à Xenomai (drivers dits « RTDM »)
  - Interface du driver et fonctions spécifiques (copies userspace/ kernelspace, timeouts...)
  - Drivers déjà existants pour certains matériels :
    - UART série 16550 (notre configuration)
    - CAN

Conception et développement du simulateur

- L'architecture dynamique pas à pas (simplifiée)



Validation du simulateur

- Validation des contraintes temporelles
  - Création du cas le plus pessimiste
    - Charge CPU au max (processus Linux)
    - Fort trafic réseau
  - Observation longue durée du PPS avec un oscilloscope numérique à mémoire (min/max PPS)
  - Tests concluants

- Intégration à l'environnement de test réel
  - Grande souplesse lors de la phase d'intégration (environnement de développement = environnement de production)

Aspects juridiques

- GPL, utilisation normale et travail dérivé
  - Utilisation normale : travail privé, pas de diffusion
  - Travail dérivé : diffusion des sources en même temps que le binaire
  - Le noyau Linux est sous licence GPL.
  - L'installation de Xenomai (sous licence GPL aussi puisque travail dérivé de Linux!) patche le noyau.
  - Quid des applications Xenomai ?
    - Elles ne constituent pas des travaux dérivés. En effet, une exception est prévue dans la licence.
      Le développpement d'application est explicitement considéré comme étant une utilisation normale.

      Il n'y a donc pas d'obligation de diffusion des sources au client final.

Bilan d'utilisation, autres applications

- Avantages
  - Aucun bug majeur rencontré
  - Coût : ni licence de développement, ni royalties lors de la diffusion
  - Démarrage rapide des projets
    - Matériel : (PC) approvisionnement en composants standards
    - Logiciel : pas de licence ni d'atelier spécifique à installer
  - Support très réactif
- Difficultés
  - Entrée dans le produit et compréhension des mécanismes : scheduling Linux, ADEOS, co-scheduling
  - Exemples d'utilisations industrielles peu nombreux
  - Profils *« Linux dév + Linux admin + temps-réel (+ métier) »* rares
- Conclusions
  - Satisfaction. Emploi prévu de Xenomai dans le cadre d'autres simulateurs spatiaux
  - Liste des matériels compatibles (avec driver RTDM) réduite, portage nécessaire si source disponible

Liens

- http://www.xenomai.org/
- http://www.c-s.fr

Droits de copie et de diffusion

- **Licence Creative Commons BY-NC-SA**

Except where otherwise noted, this work is licensed under
http://creativecommons.org/licenses/by-nc-sa/3.0/

- **Paternité-Pas d'Utilisation Commerciale-Partage des Conditions Initiales à l'Identique 2.0**
  - Vous êtes libres :
  - de reproduire, distribuer et communiquer cette création au public
  - de modifier cette création
  - Selon les conditions suivantes :
  - **Paternité**. Vous devez citer le nom de l'auteur original de la manière indiquée par l'auteur de l'oeuvre ou le titulaire des droits qui vous confère cette autorisation (mais pas d'une manière qui suggérerait qu'ils vous soutiennent ou approuvent votre utilisation de l'oeuvre).
  - **Pas d'Utilisation Commerciale**. Vous n'avez pas le droit d'utiliser cette création à des fins commerciales.
  - **Partage des Conditions Initiales à l'Identique**. Si vous modifiez, transformez ou adaptez cette création, vous n'avez le droit de distribuer la création qui en résulte que sous un contrat identique à celui-ci.
  - A chaque réutilisation ou distribution de cette création, vous devez faire apparaître clairement au public les conditions contractuelles de sa mise à disposition. La meilleure manière de les indiquer est un lien vers cette page web.
  - Chacune de ces conditions peut être levée si vous obtenez l'autorisation du titulaire des droits sur cette oeuvre.
  - Rien dans ce contrat ne diminue ou ne restreint le droit moral de l'auteur ou des auteurs.