Gestion de la réactivité des communications réseau

François Trahay

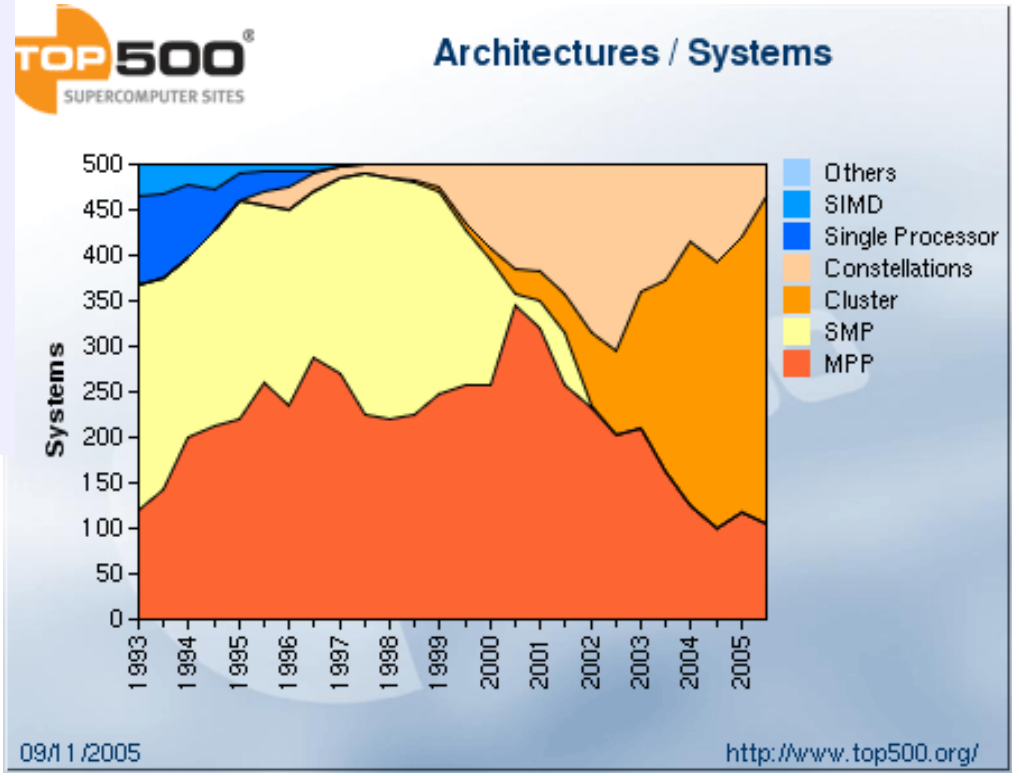
Runtime, LaBRI

sous la direction d'Alexandre Denis

Université Bordeaux I

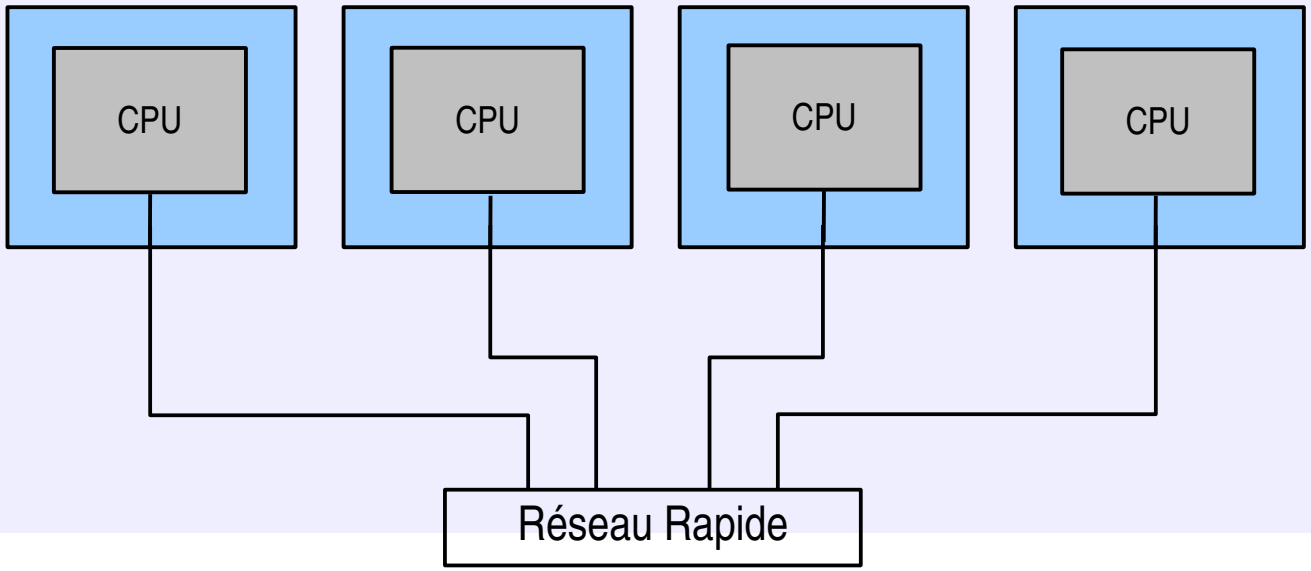
Le calcul hautes performances

- La tendance actuelle : les grappes de machines standard
  - bon rapport coût / performance
- Caractéristiques
  - Réseaux rapides
    - Latence ~ 1µs
    - Débit ~ 10 Gb/s
  - Nœuds multiprocesseurs

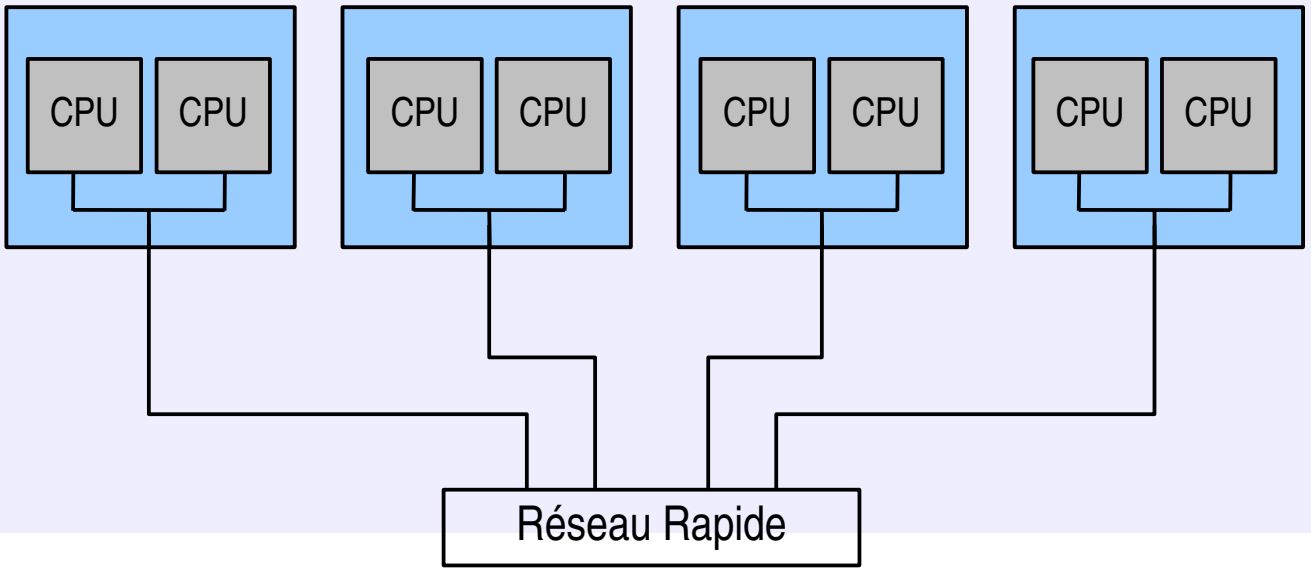


Evolution du Top 500 de 1993 à 2005

Hiérarchisation des architectures

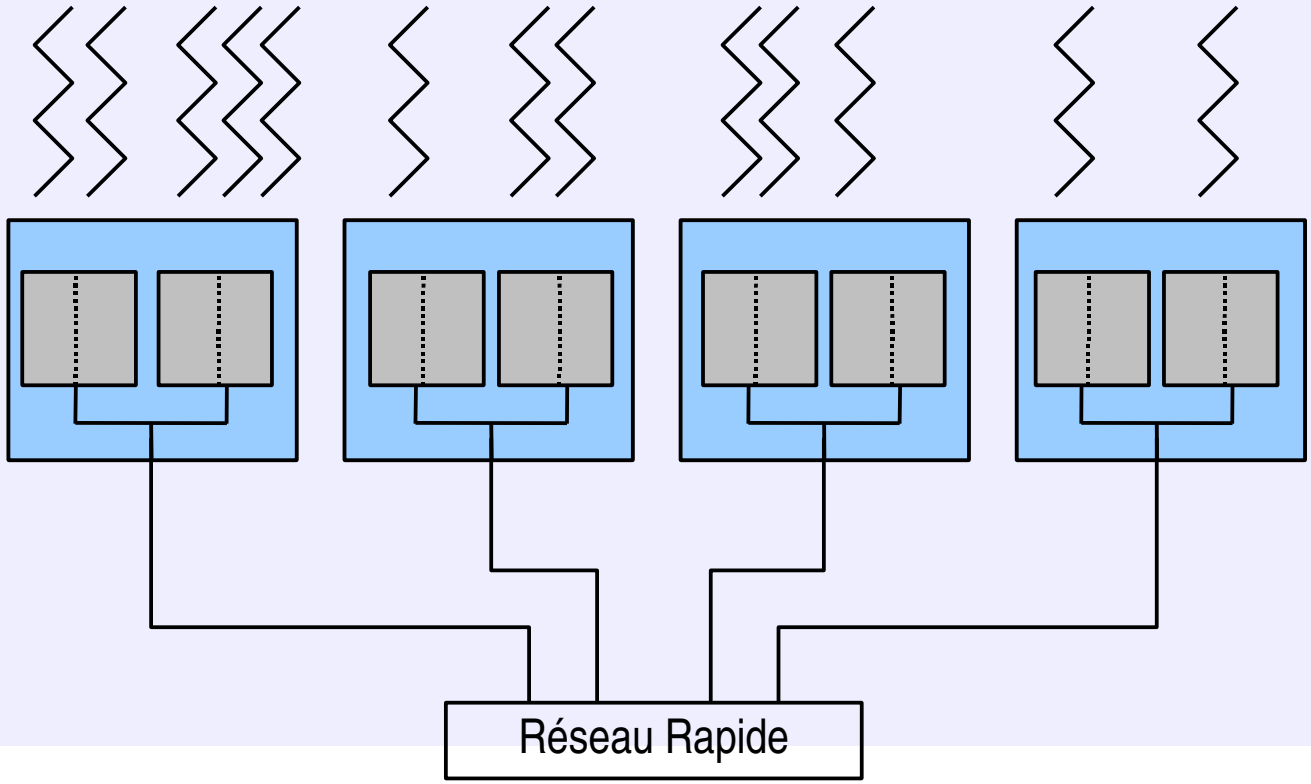


Hiérarchisation des architectures



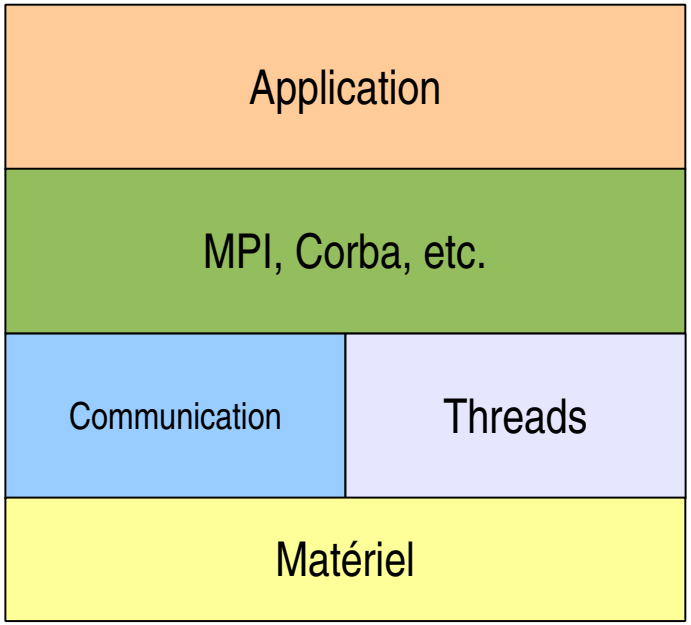
Hiérarchisation des architectures

Comment exploiter ces architectures?



Programmation

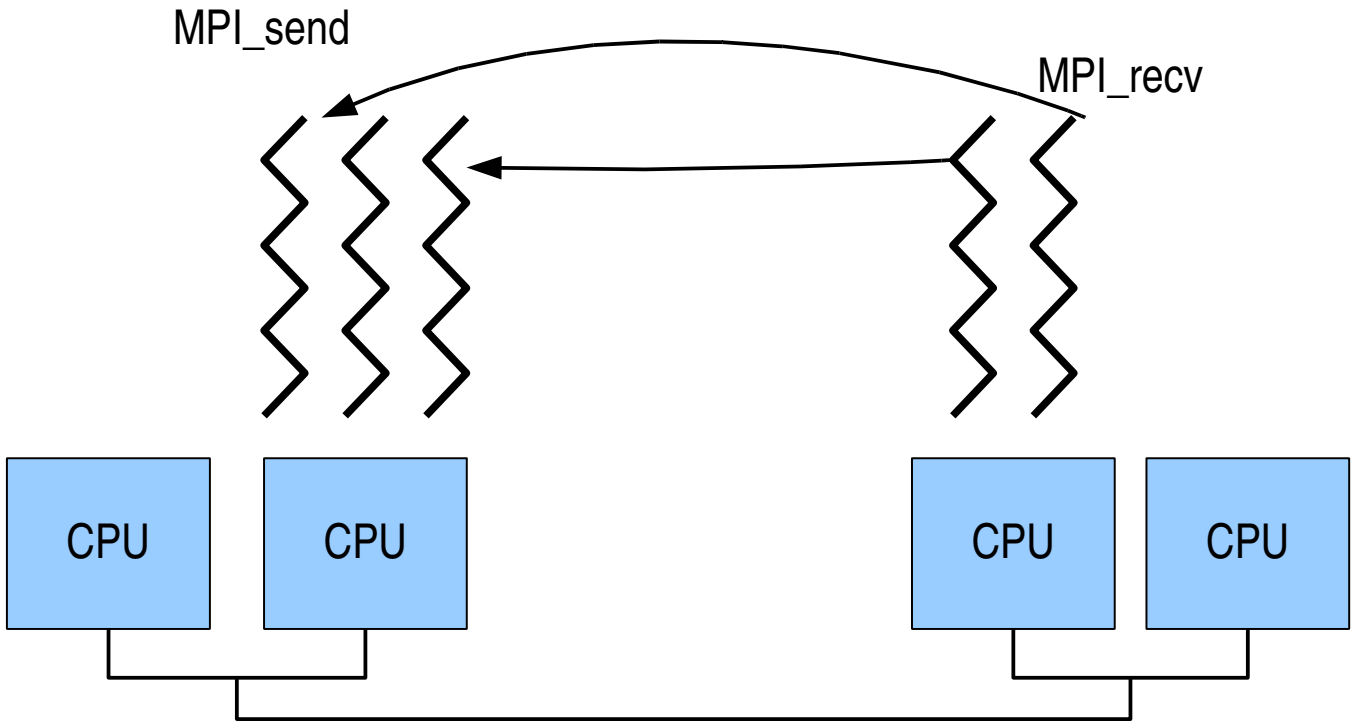
- Importance de la portabilité
  - Ne pas réécrire un code qui fonctionne
- Standardisation des environnements de programmation
  - MPI
  - OpenMP
- Environnements reposant sur des supports d'exécution
  - Bibliothèques de threads
    - Utilisation d'ordonnanceur à deux niveaux
  - Bibliothèque de communication



Exemple de pile logicielle utilisée sur une grappe

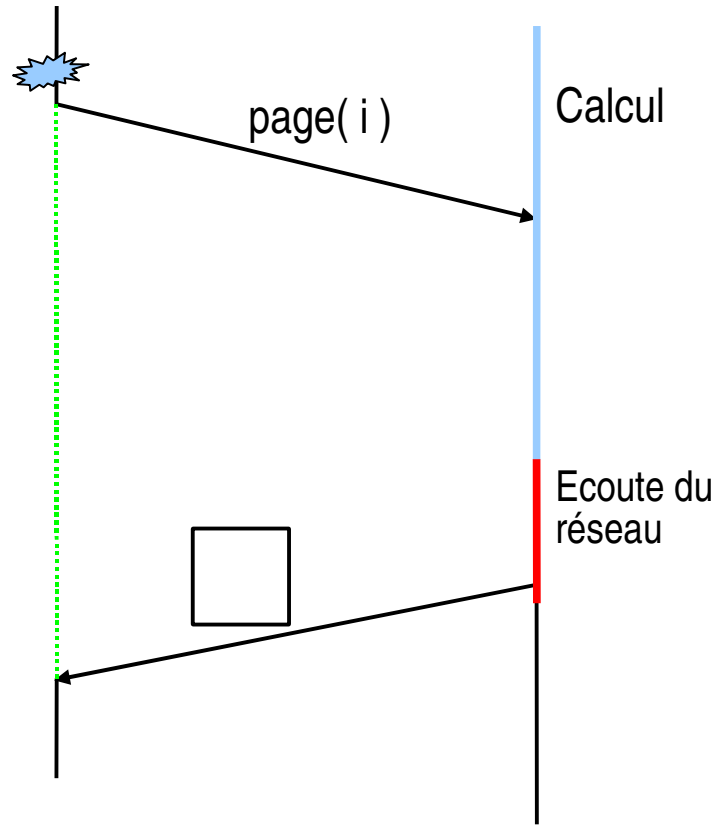
L'importance des communications

- Multiples communications entre threads sur des noeuds différents
- Certaines communications nécessitent des “temps de réponse” courts !



Exemple de schéma de communication « interactif»

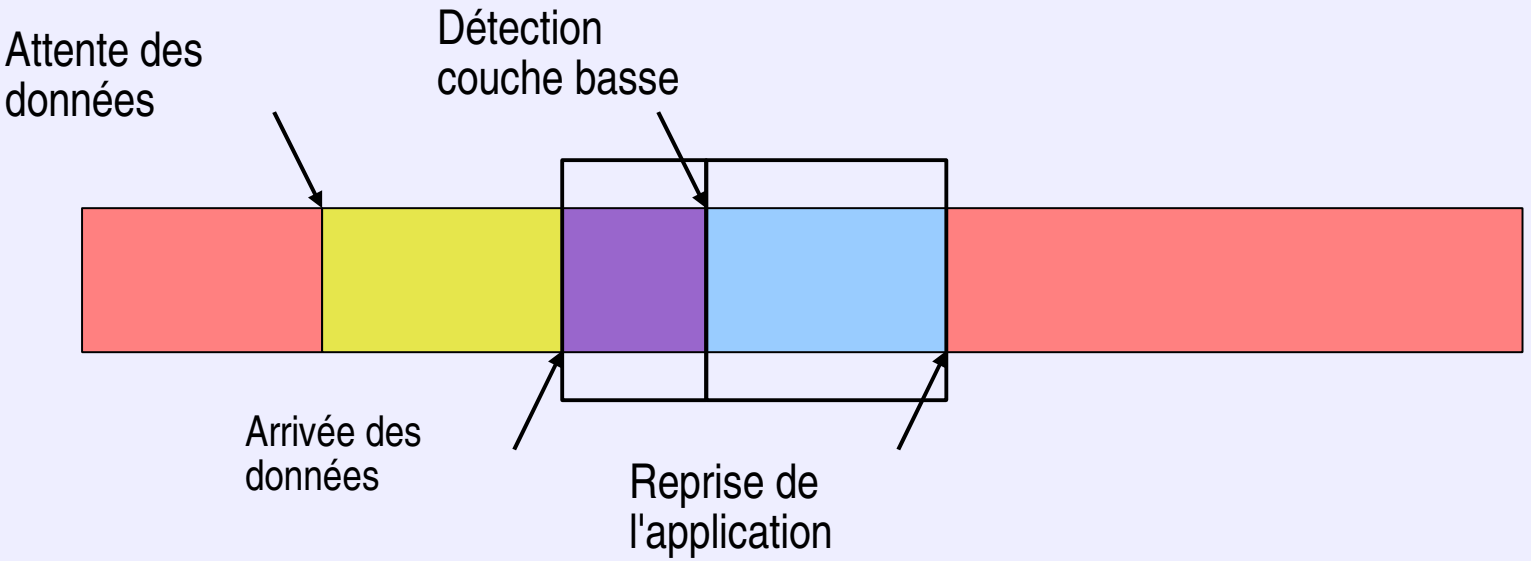
- DSM (Mémoire Partagée Distribuée)
  - Requête de demande de page suite à un défaut de page



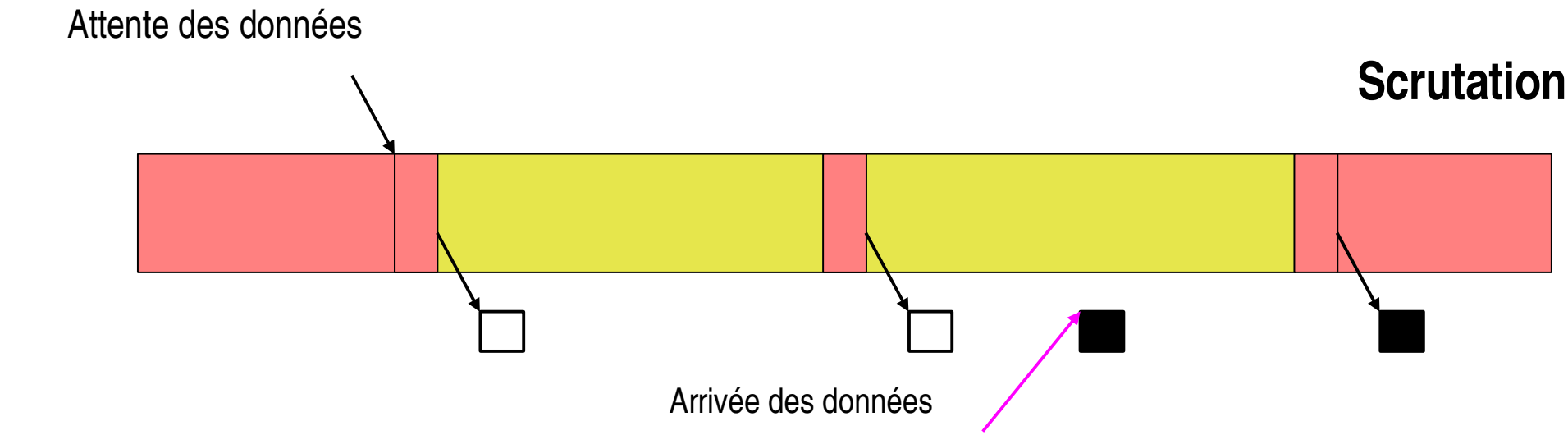
Influence du temps de réponse dans une DSM

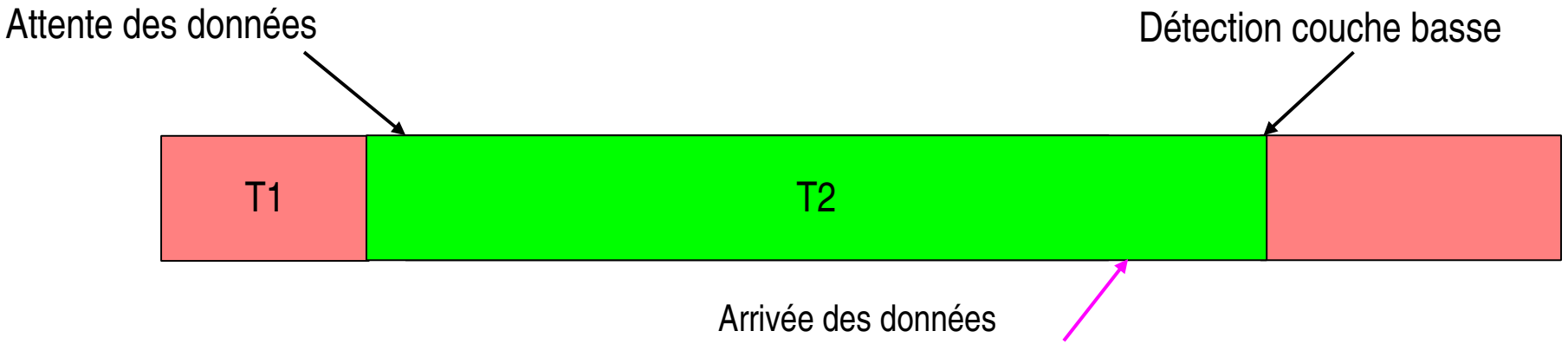
La réactivité

- Capacité à prendre en compte rapidement un événement

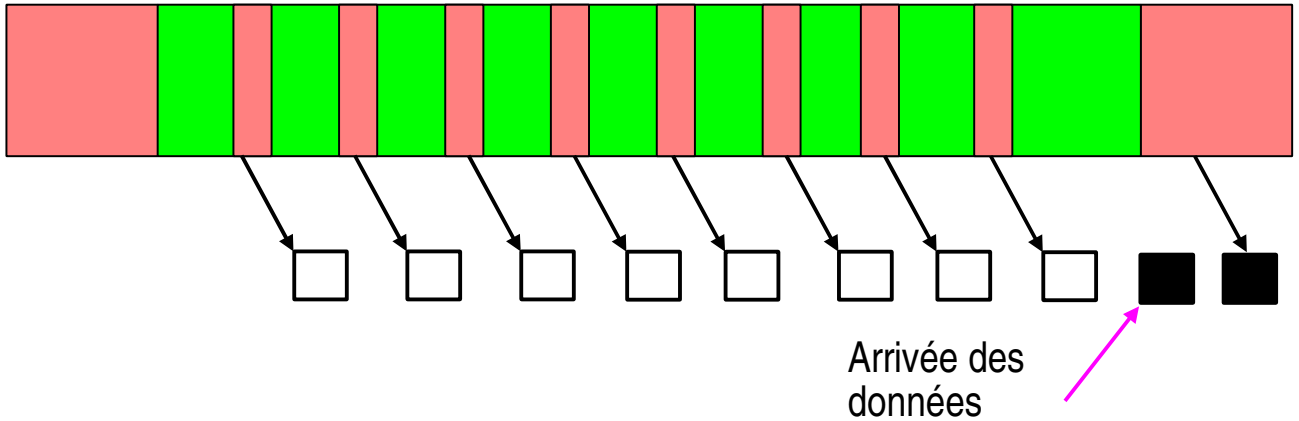


Détection des événements : 2 mécanismes

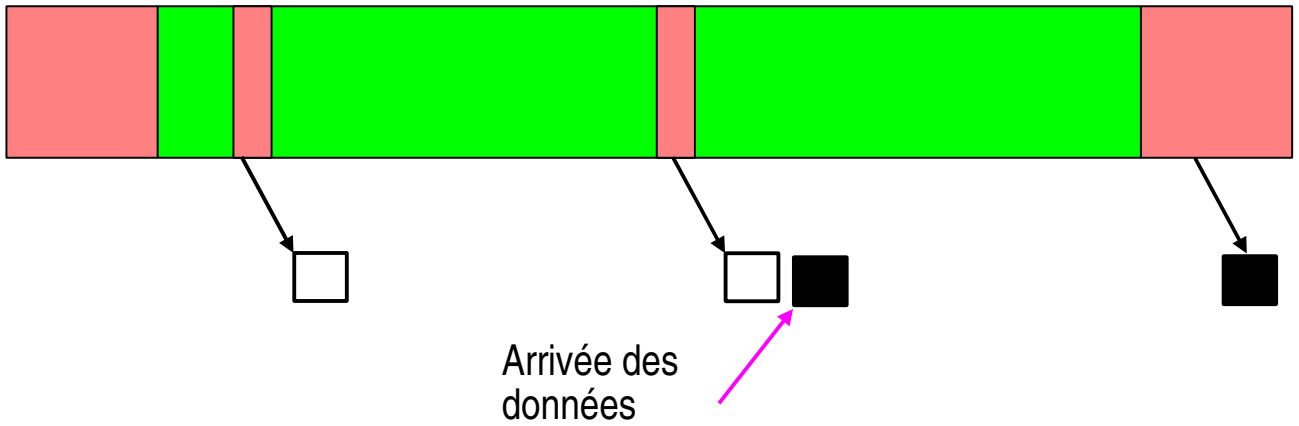




Problème de scrutation



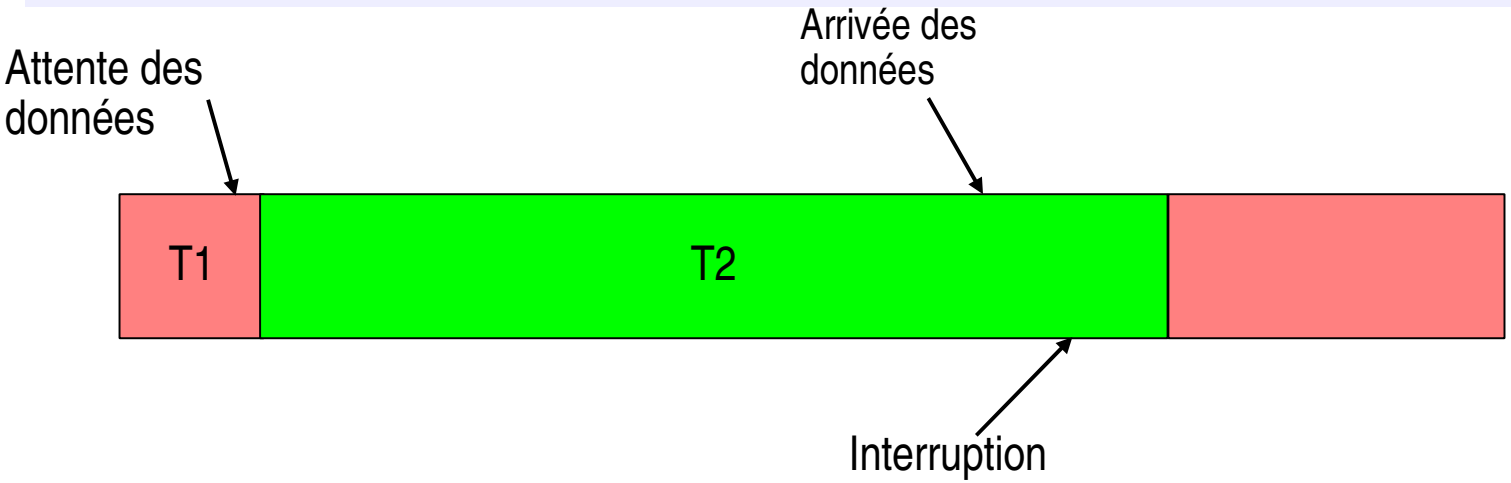
Fréquence trop élevée



Fréquence trop basse

Interruptions

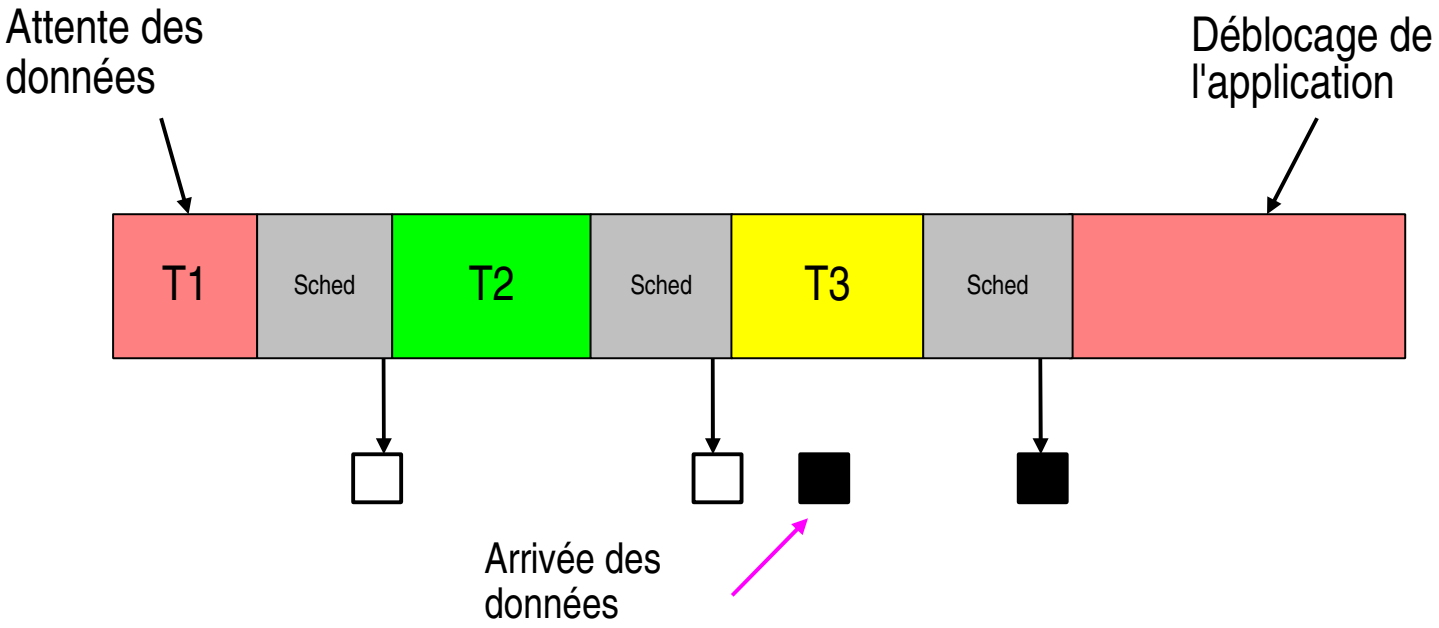
- Interruption :
  - Interruption générée par la carte réseau
  - Problème:
    - Coût de l'interruption



Serveur d'événements

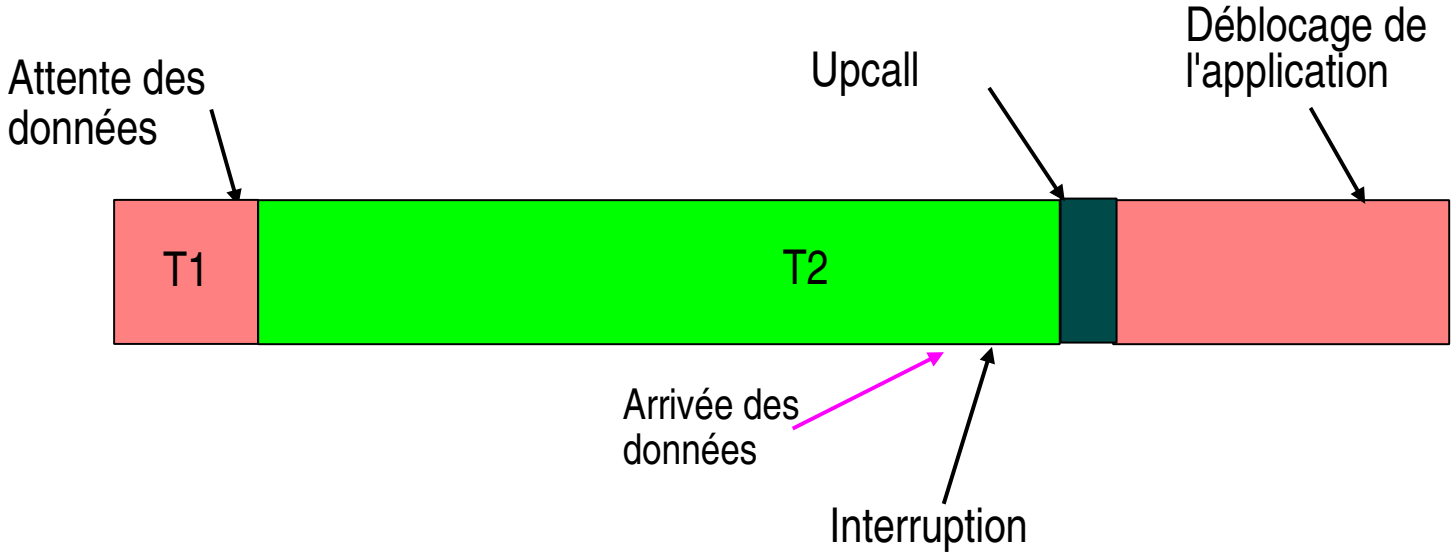
Serveur d'événements (Danjean, 2004)

- Intégrer la scrutation à l'ordonnanceur
- Réactivité **bornée** à *quantum*
- Réactivité moyenne de *quantum* / 2

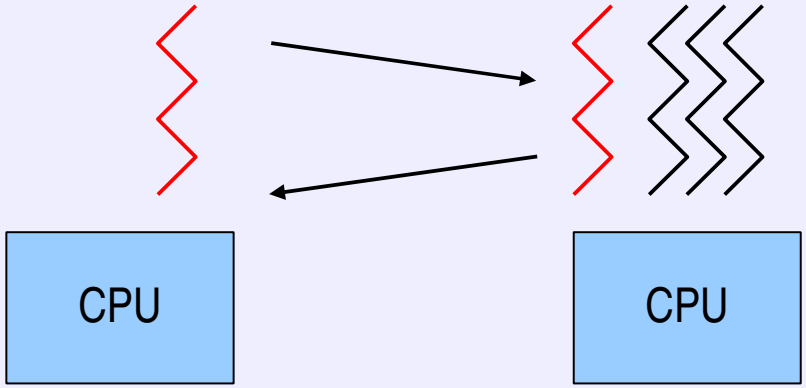


Interruptions au niveau utilisateur : les Activations

- Scheduler Activations (Anderson, 1990)
  - Ne pas bloquer le thread noyau complet
  - Utilisation d'***upcall*** pour réveiller le thread bloqué
  - Effondrement des temps de réaction quand le nombre de requêtes augmente
  - **Nécessité une modification lourde du noyau**



Ping Pong avec charge

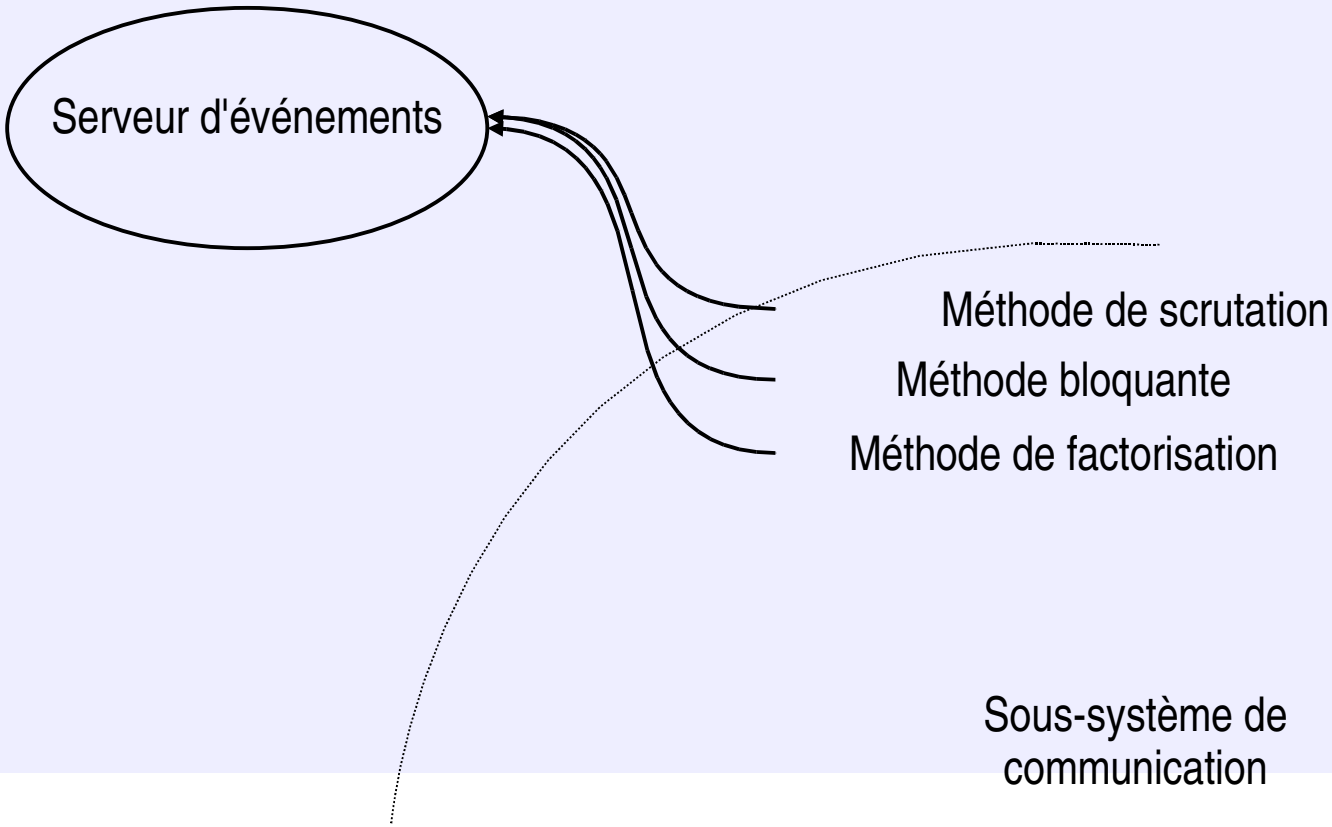


	Nombre de threads de calcul				
Version de l'ordonnanceur	Aucun	1	2	5	10
Scrutation naïve (ms)	**0,13**	5	10	25	**50**
Serveur d'événements (ms)	**0,13**	4,84	4,84	5,47	**4,85**
Activations (ms)	**0,45**	0,45	0,45	0,45	**0,45**

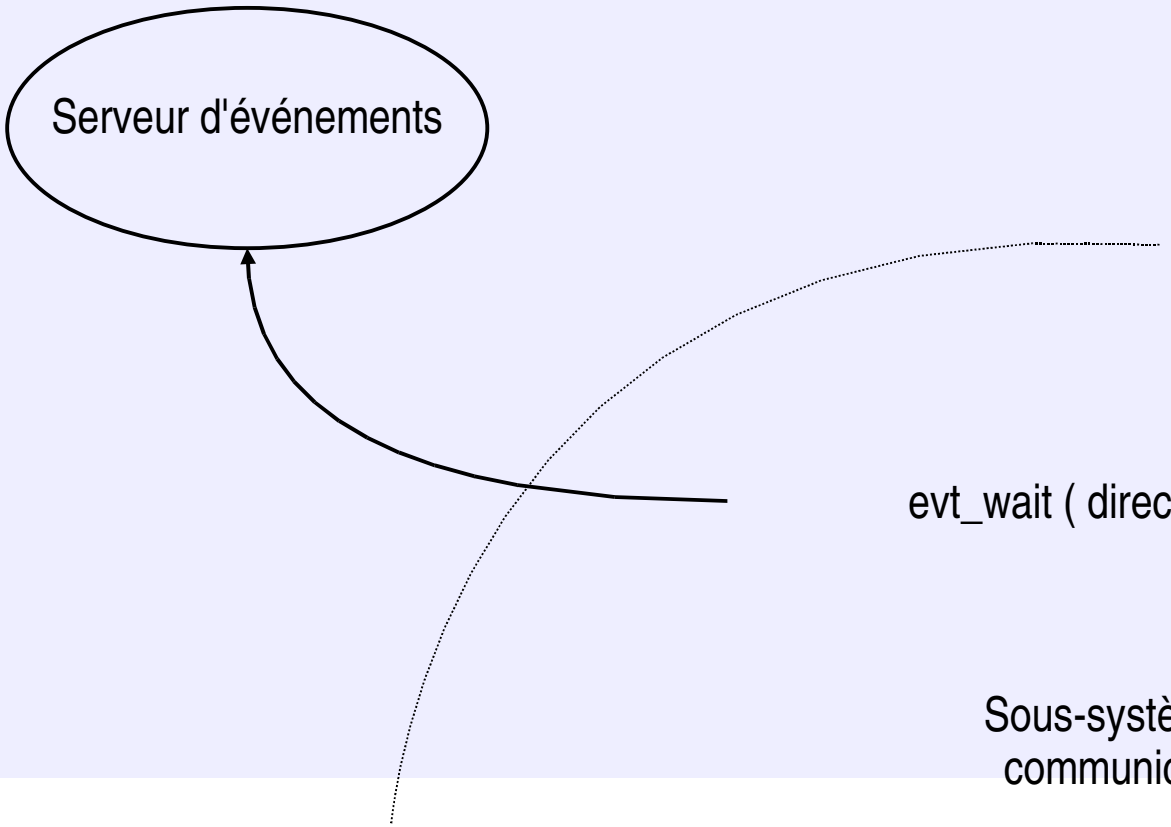
Synthèse

- Utilisation « intelligente » de la scrutation et des interruptions
  - Utiliser la méthode la plus adaptée
  - Inspiré de Panda (Langendoen, 1996)
- Centralisation des requètes
  - Inspiré du serveur d'événements de Danjean
- Factorisation de requètes
  - Eviter les scrutations multiples

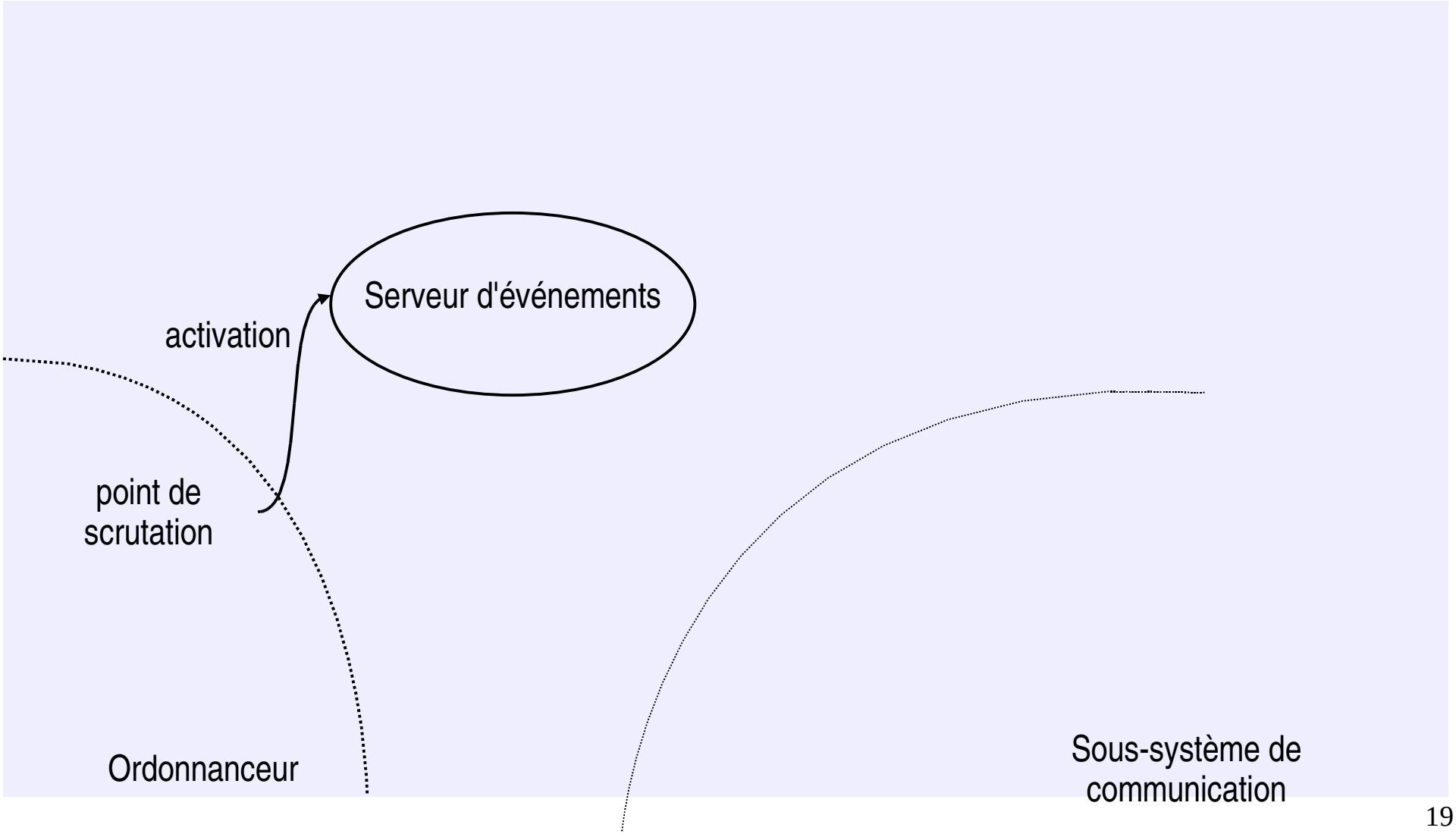
Contribution : un serveur d'événements « intelligent »



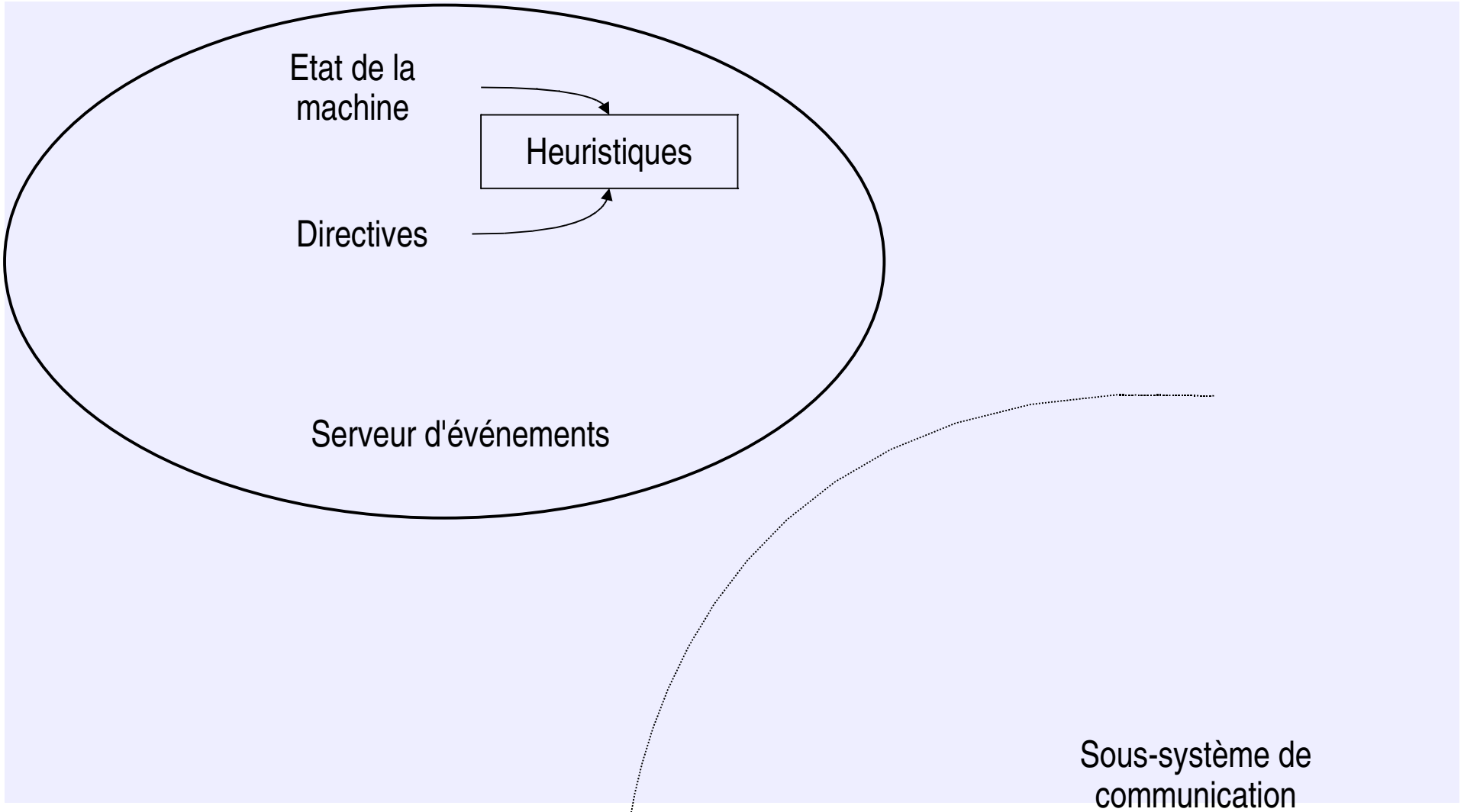
Contribution : un serveur d'événements « intelligent»



Contribution : un serveur d'événements « intelligent»



Contribution : un serveur d'événements « intelligent»



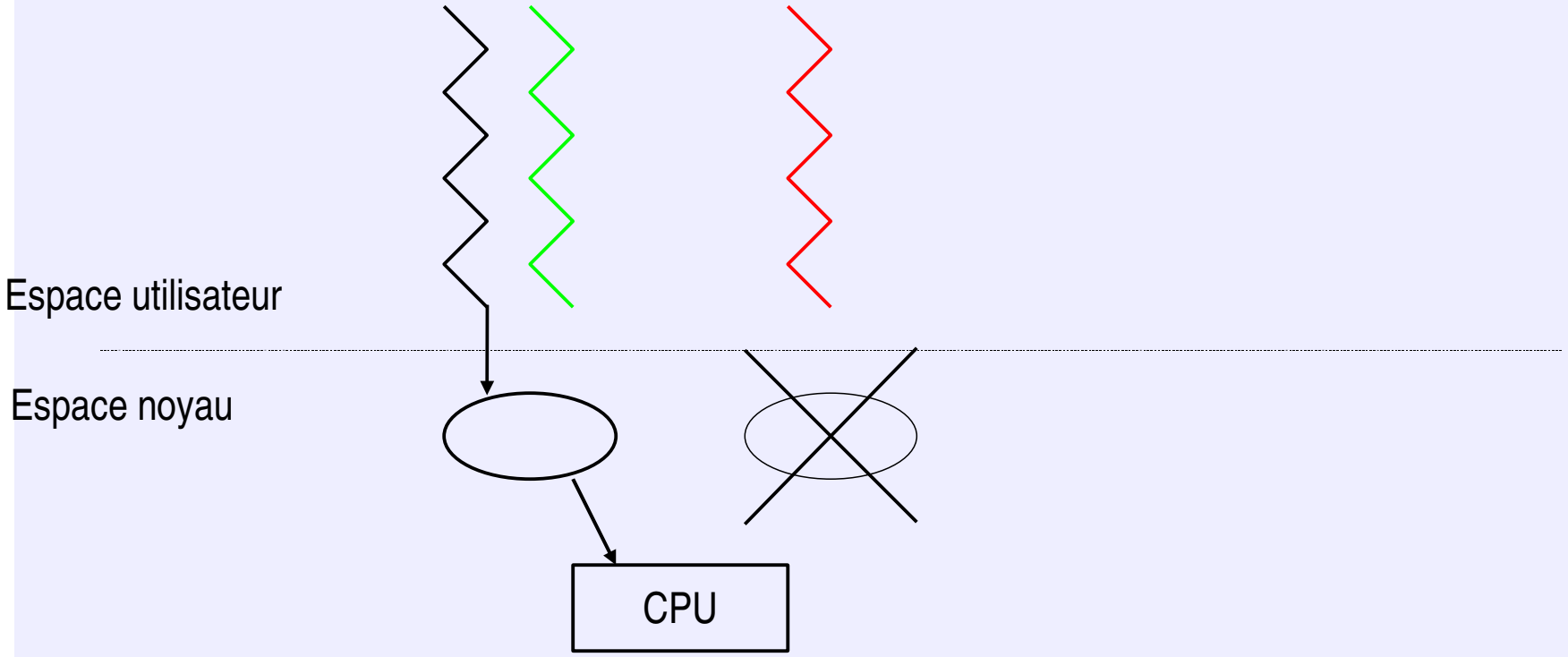
Implémentation

- Intégration dans la suite logicielle PM2
  - Madeleine : bibliothèque de communication
  - Marcel : bibliothèque de threads
  - PadicoTM : environnement de communication pour les grilles
- Implémentation d'un appel bloquant adapté aux threads utilisateurs
- Reste à rendre le serveur d'événements « intelligent »

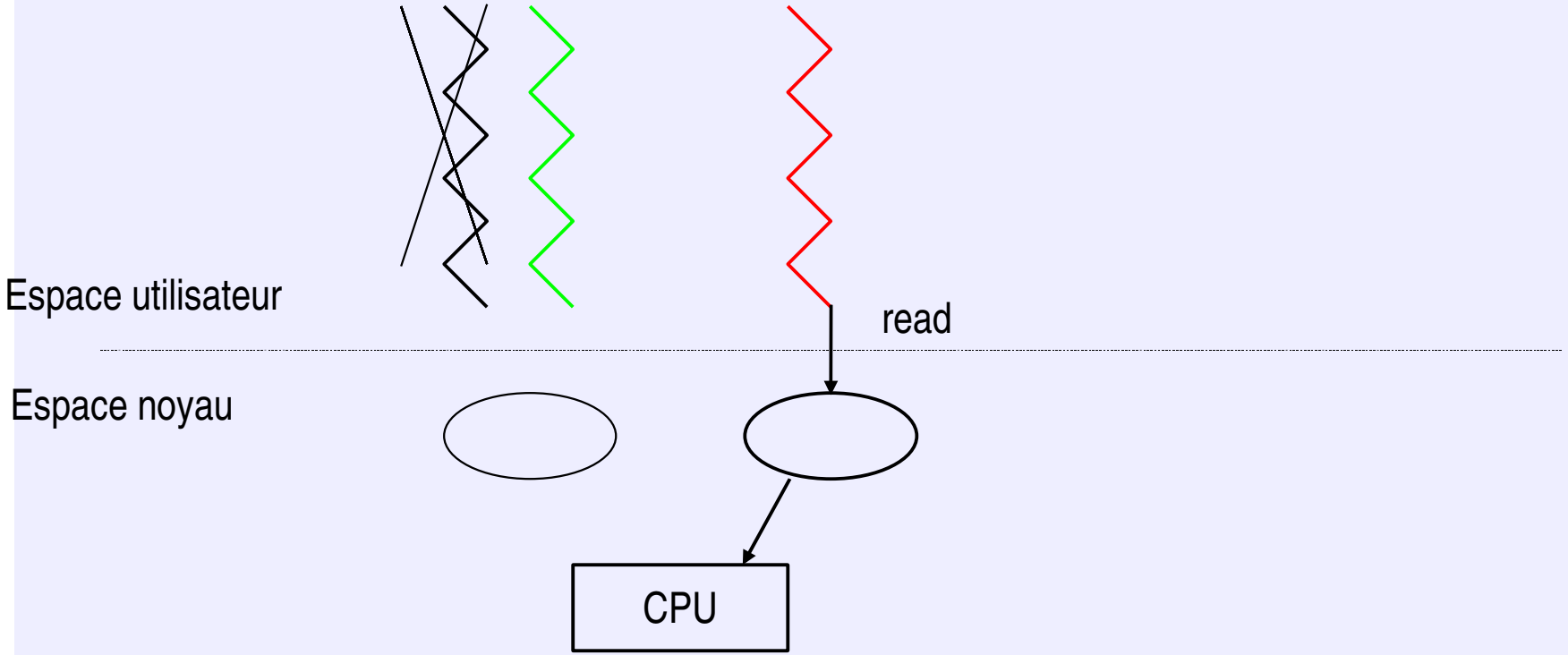
Gestion des appels bloquants

- Problème le plus important rencontré
- Appels bloquants et multithreading de niveau utilisateur
  - Un appel bloquant suspend le thread noyau sous-jacent
- Méthode proposée :
  - Isolation des appels bloquants sur un autre thread noyau
    - Conserve la réactivité des appels bloquants
    - Conserve la légèreté de la scrutation

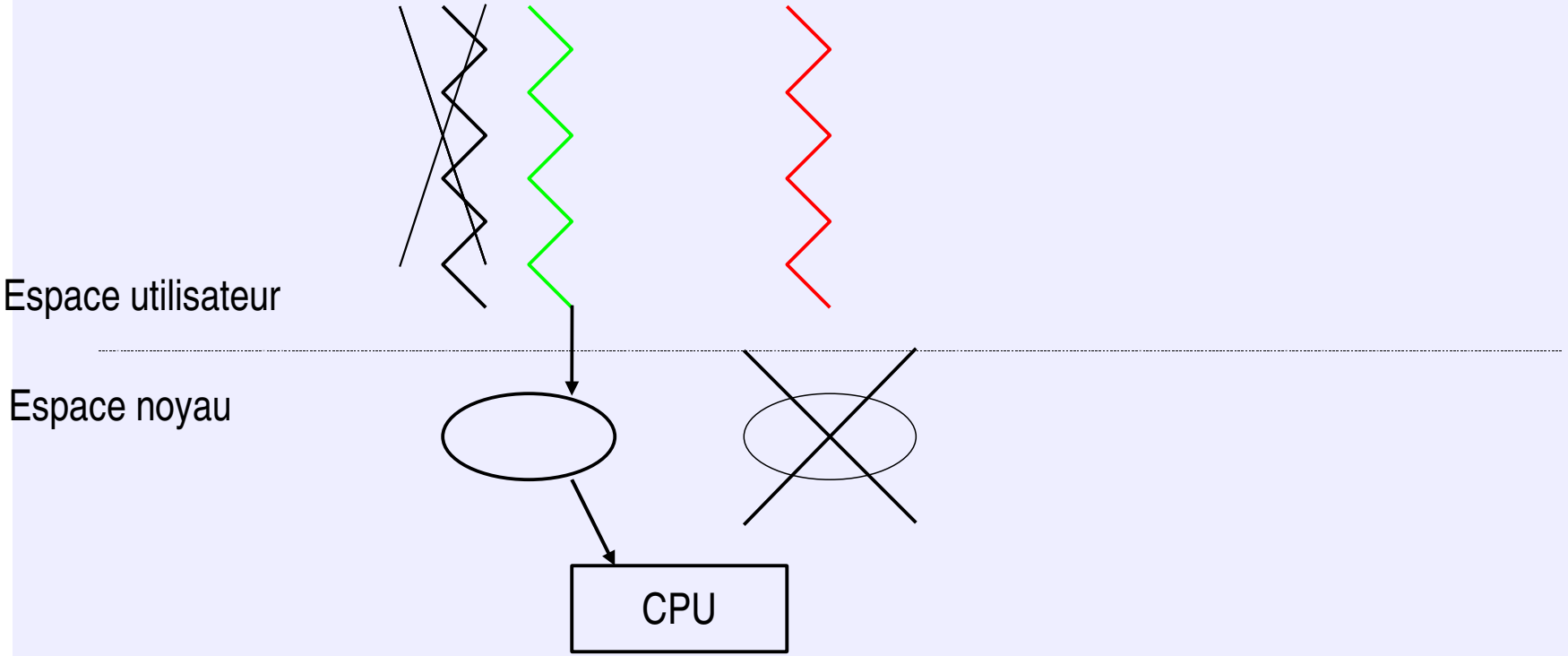
Demande de réception



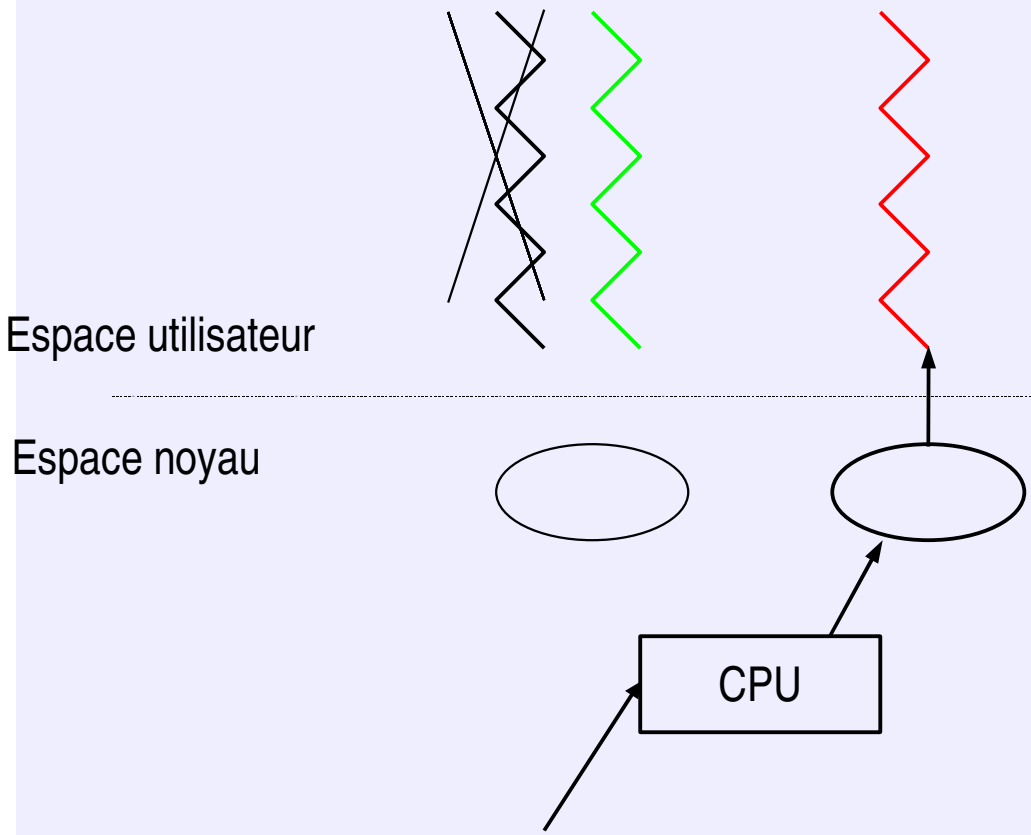
Appel système déporté



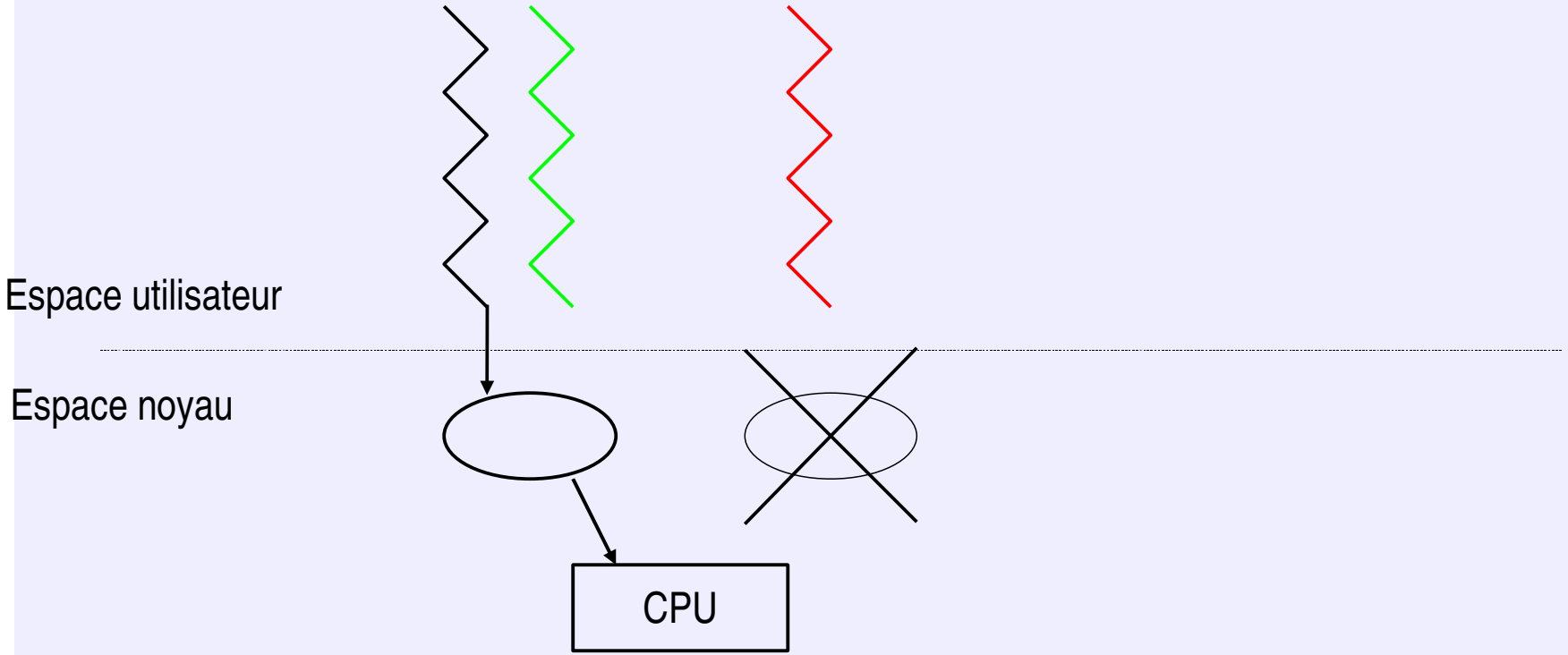
Retour au thread de calcul



Arrivée de la communication



Retour au thread utilisateur



Evaluation

- Environnement de test :
  - Machines NUMA (4 cores à 1,8 Ghz)
  - TCP/GigaEthernet 1Gb/s
- Ping pong sur des machines chargées

Réactivité / nombre de threads

Méthode utilisée	Nombre de threads de calcul			
	0	1	2	3
Scrutation naïve (ms)	0,04	5,0	10,0	15,0
Serveur d'événements (ms)	0,04	5,5	5,5	5,5
Appel système déporté (ms)	0,05	0,05	0,05	0,05

Surcoût de l'exportation

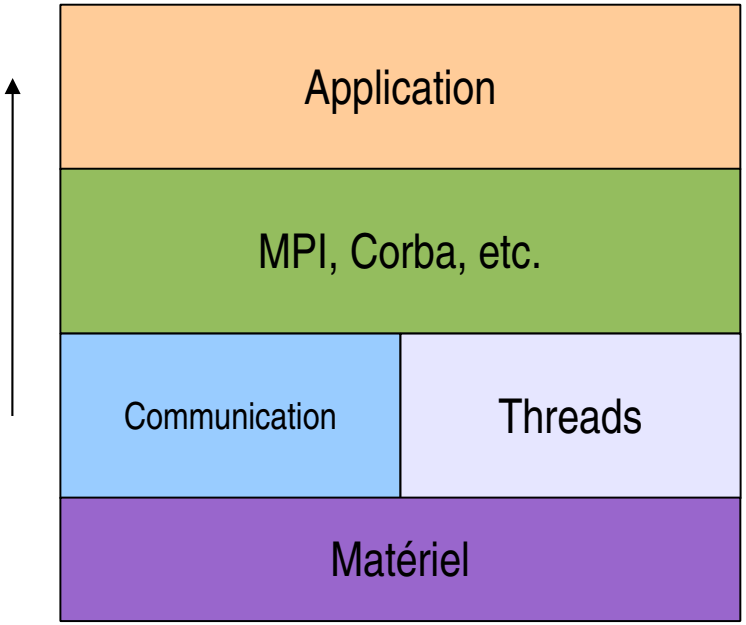
- Surcoût recouvrable (au pire cas) : 6 µs
  - Réveil du thread de communication
- Surcoût non recouvrable (au pire cas) : < 9 µs
  - Déblocage du thread utilisateur
  - Mise en attente du thread de communication

Conclusion

- Travail réalisé
  - Etude des schémas de communication
  - Généralisation des travaux précédents (Panda, Serveur d'événements)
  - Elaboration d'un serveur d'événements “intelligent”
- Perspectives
  - Migration de threads
  - Mise en place du serveur d'événements
  - Réactivité du système entier

Et après la réception ?

- Comment transmettre l'information jusqu'au thread applicatif ?
  - Nombreux threads impliqués
  - Threads de calcul perturbateurs
  - Problème de réentrance



Décomposition d'un ping pong