Génération automatique d'implémentation distribuée à partir de modèles formels de processus concurrents asynchrones

Hugues Evrard

Université Grenoble Alpes

CONVECS – Inria LIG

Systèmes concurrents

NoC

multi-core

cluster

cloud

- Concurrence : plusieurs fils d'exécutions
  - systèmes parallèles / distribués
  - exécution **non déterministe**
  - complexité : développement, test, maintenance ...

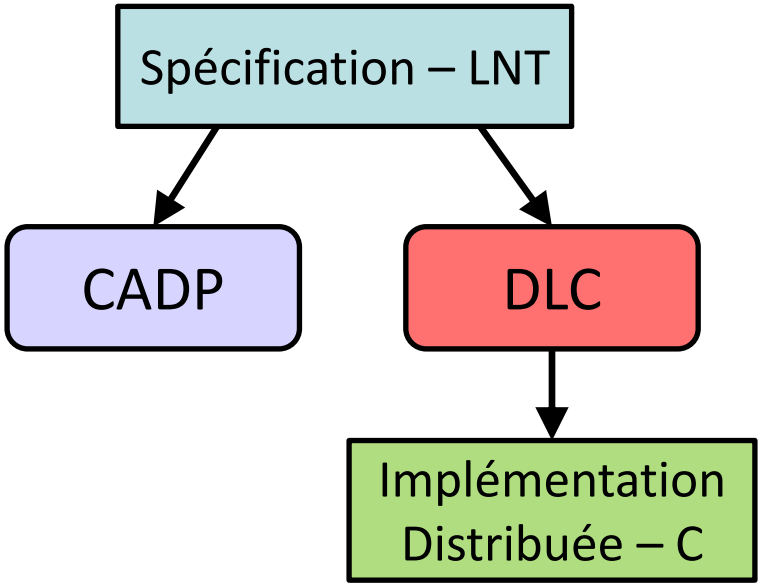
Méthodes formelles

- Étude mathématique de systèmes
  - preuve, interprétation abstraite, …
- Concurrence : **algèbres de processus**
  - processus + interactions
  - CCS, CSP, π-calcul, LOTOS, …
- Approche formelle
  - **modèle** en langage formel
  - **vérification** du modèle
  - **implémentation** : si à la main, introduction bogues ?
  - **génération automatique** d'implémentation

CADP

- ***C****onstruction and* ***A****nalysis of* ***D****istributed* ***P****rocesses*
- Boîte à outils formels (50+ outils)
  - langage de modélisation : **LNT**
  - *model-checking* (propriétés : **MCL**)
  - *equivalence-checking*
  - génération de code séquentiel : **EXEC/CAESAR**
- Vérification de système distribué …
  - … génération d'implémentation distribuée ?

Thèse – vue d'ensemble



- But : **cadre formel** de dév. de **systèmes distribués**
  - vérif. et implém. à partir du même modèle formel
- Nouvel outil : **DLC – *Distributed LNT Compiler***

Plan

- Introduction
- Le langage LNT
- Vérification formelle de protocole
- Protocole de rendez-vous
- Fonctions crochets
- Architecture DLC
- Expérimentations
- Conclusion

Le langage LNT

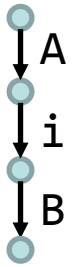
LNT

- Types de données abstraits
  - prédéfinis : bool, nat, int, real, char, string
- Fonctions
  - if … then … else … end if
  - for … while … by … loop … end loop
  - *pattern matching* : case … in … -> … | … end case
- Processus
  - sur-ensemble des fonctions
  - effectue des actions : sur porte, ou interne

LNT – Processus

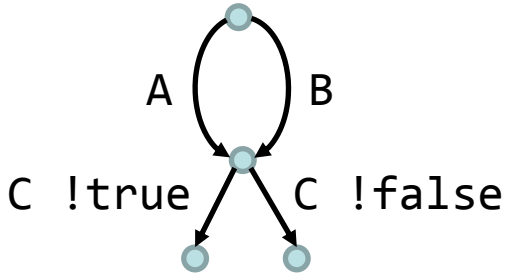
- Sémantique **LTS** (*Labelled Transition System*)
  - espace d'états du système

```
process P1 [A,B] is
   A ; i ; B
end process
```



- Non-déterminisme : select, réception de donnée

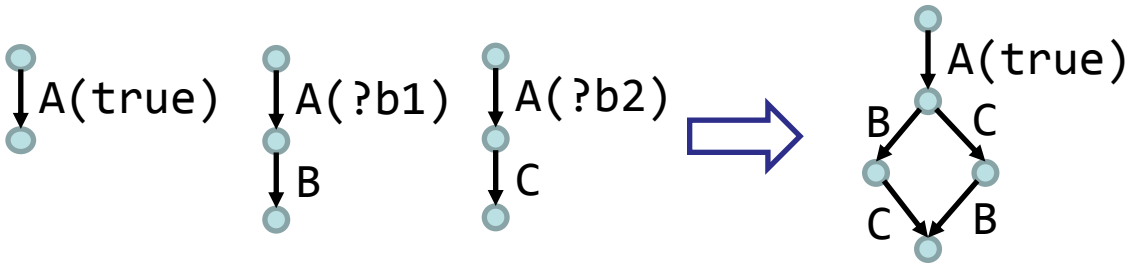
```
process P2 [A,B,C] is
   var b: bool in
      select
         A
      [] B
      end select ;
      C(?b)
   end var
end process
```



LNT – Interactions

- Processus interagissent par rendez-vous multiple
  - synchronisation lors d'une action sur porte
  - multiple : *n* processus concernés (*n* >= 2)
  - possible échange de données par des offres (envoi/réception)
- Composition parallèle : fixe les rendez-vous possibles
  - sémantique entrelacement : non-déterminisme

```
par A in
   A(true)
|| A(?b1) ; B
|| A(?b2) ; C
end par
```



LNT – Rendez-vous

- Mélange d'offres envoi / réception

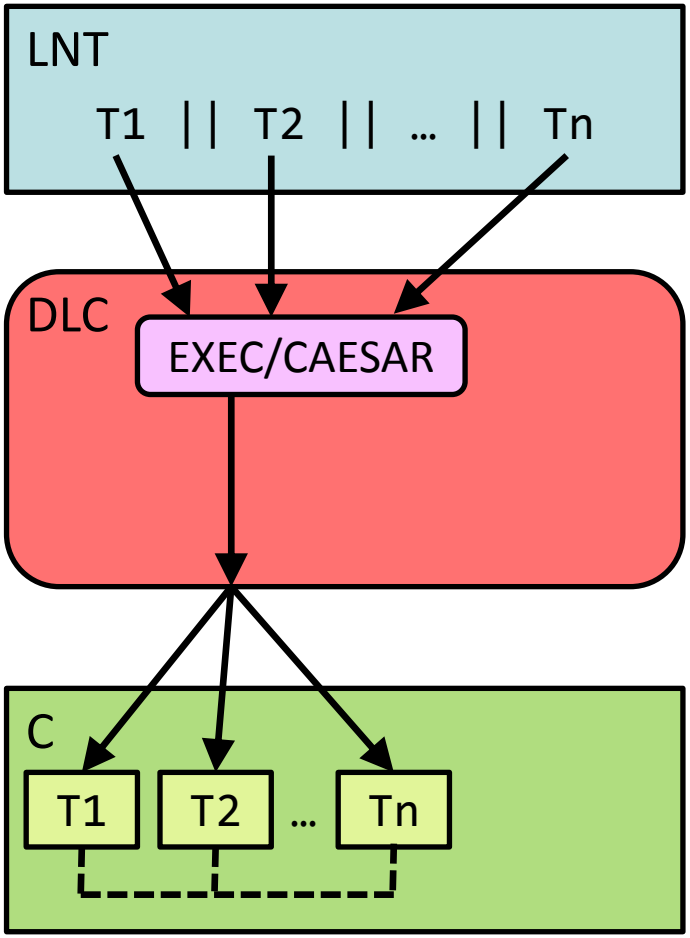
```
par A in
   A (?n, true)
|| A (42, ?b)
end par
```

- Opérateur n-parmi-m [Garavel-Sighireanu-99]

```
par A, B #2 in
   C -> P1[A,B,C]
||      P2[A,B]
|| C -> P3[A,B,C]
end par
```

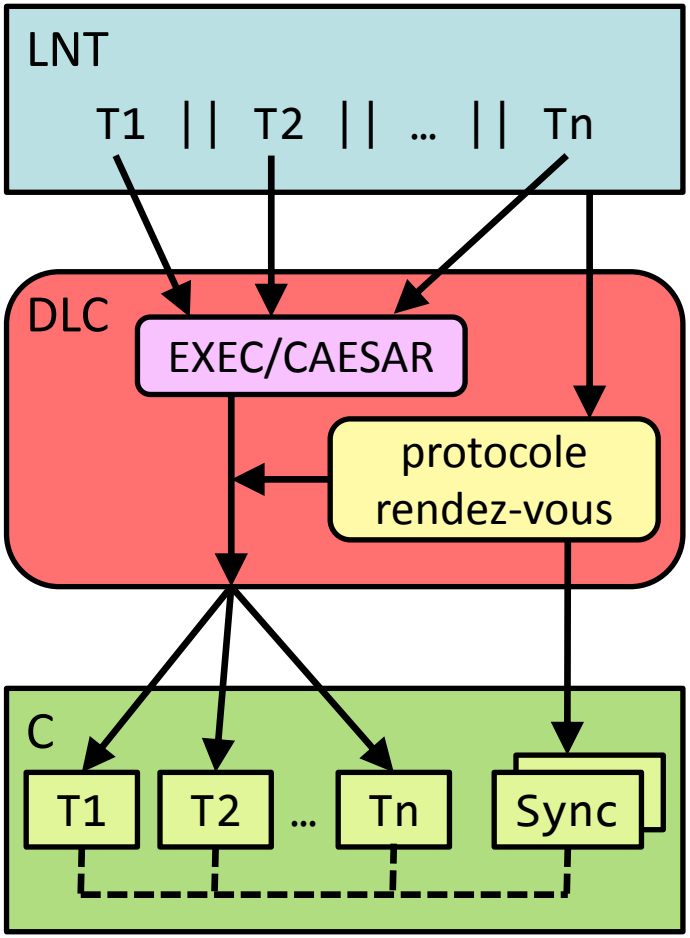
- Représentation : vecteurs de synchronisation
  - A : (P1,P2,P3)
  - B : (P1,P2), (P1,P3), (P2,P3)
  - C : (P1,P3)
- Rendez-vous en conflit lorsque processus en commun

Implémentation distribuée



- Tâche : programme séquentiel
  - généré par EXEC/CAESAR
  - code à compléter
- Communication : TCP
  - asynchrone, ordonné

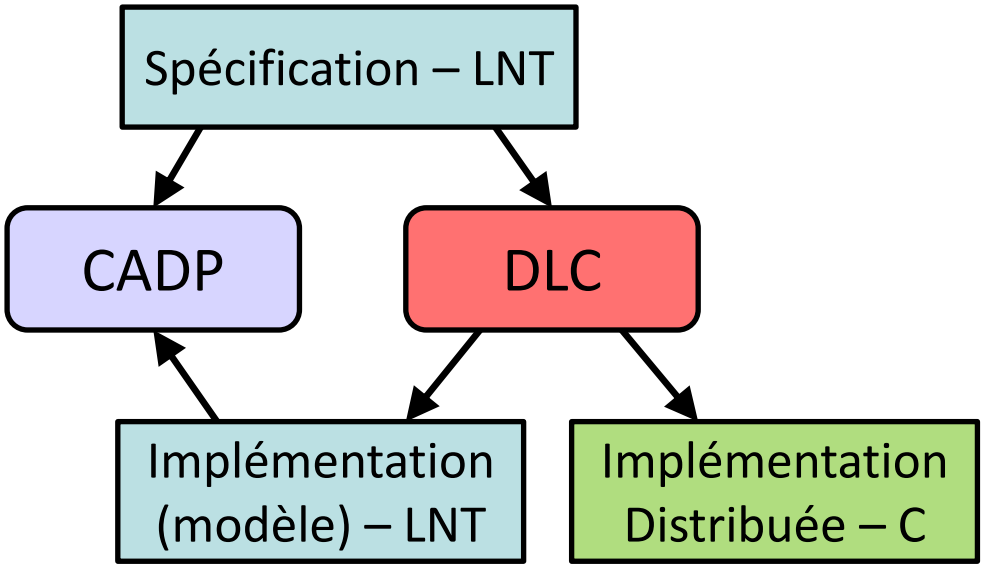
Implémentation distribuée



- Tâche : programme séquentiel
  - généré par EXEC/CAESAR
  - code à compléter
- Communication : TCP
  - asynchrone, ordonné
- Interaction : protocole de rdv
  - exclusion mutuelle des conflits
  - processus aux. synchroniseurs
- Protocole efficace et correct
  - vérification formelle

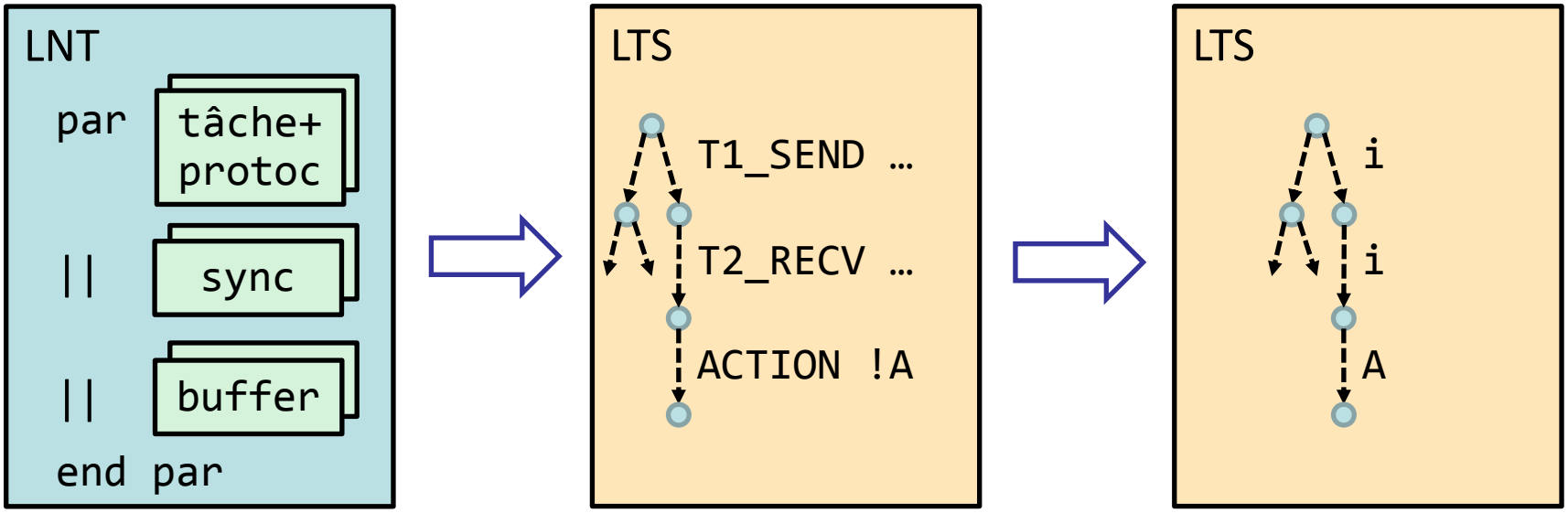
Vérification formelle de protocole de rendez-vous

Méthode de vérification



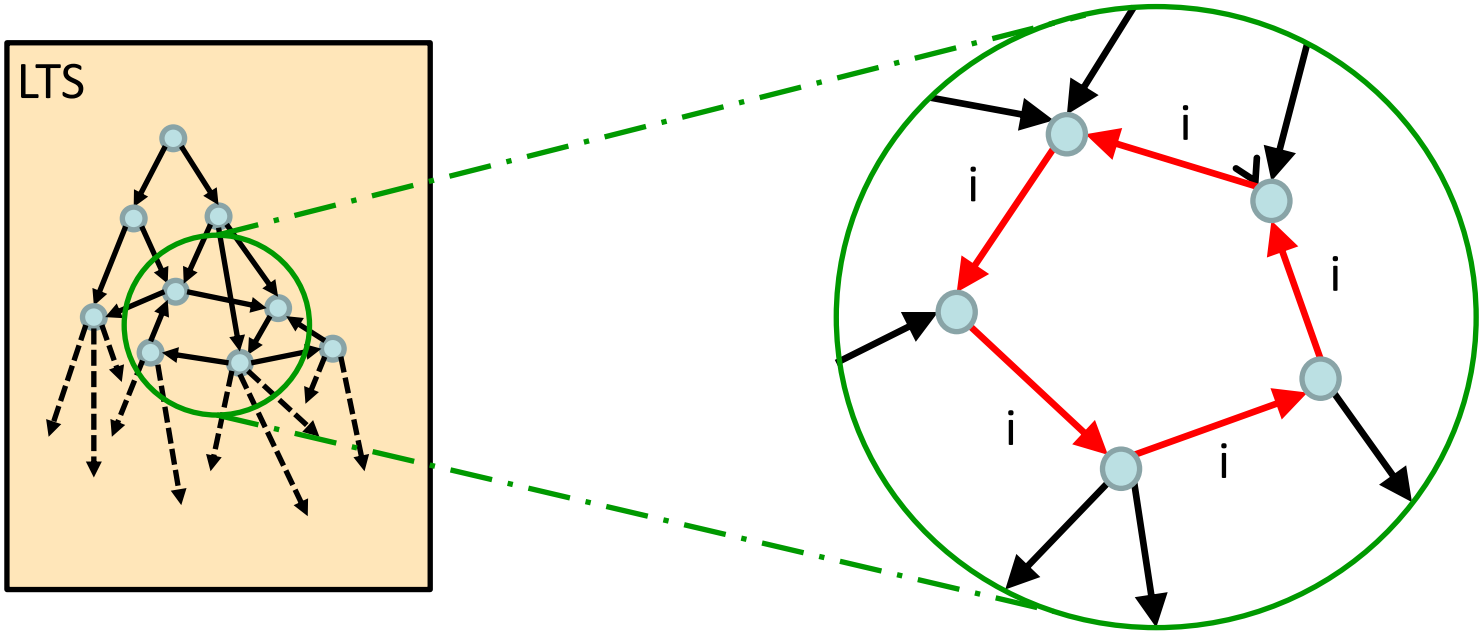
- Modèle formel de l'implémentation
- Vérifications en 3 temps :
  - absence livelock
  - absence deadlock
  - équivalence entre spécification et implémentation

Modèle de l'implémentation



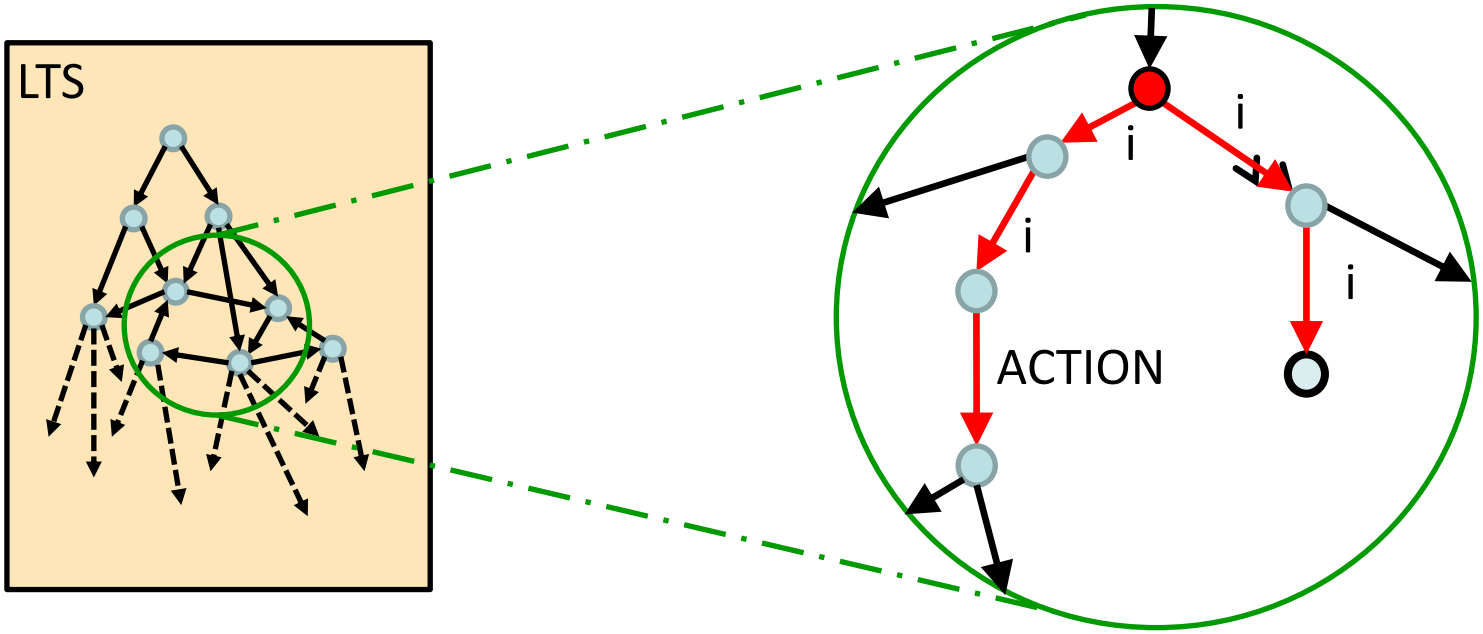
- Modèle de messages asynchrones : buffers
- 2 types d'étiquettes : message protocole, action de spec.
- Préparation pour comparaison avec spécification
  - dissimulation : message protocole → action interne
  - renommage : annonce d'action → action niveau specification

Absence de livelock



- Négociation infinie sans aboutir à une action
  - LTS : boucle d'actions internes
  - propriété MCL : `<true*> <"i">@`

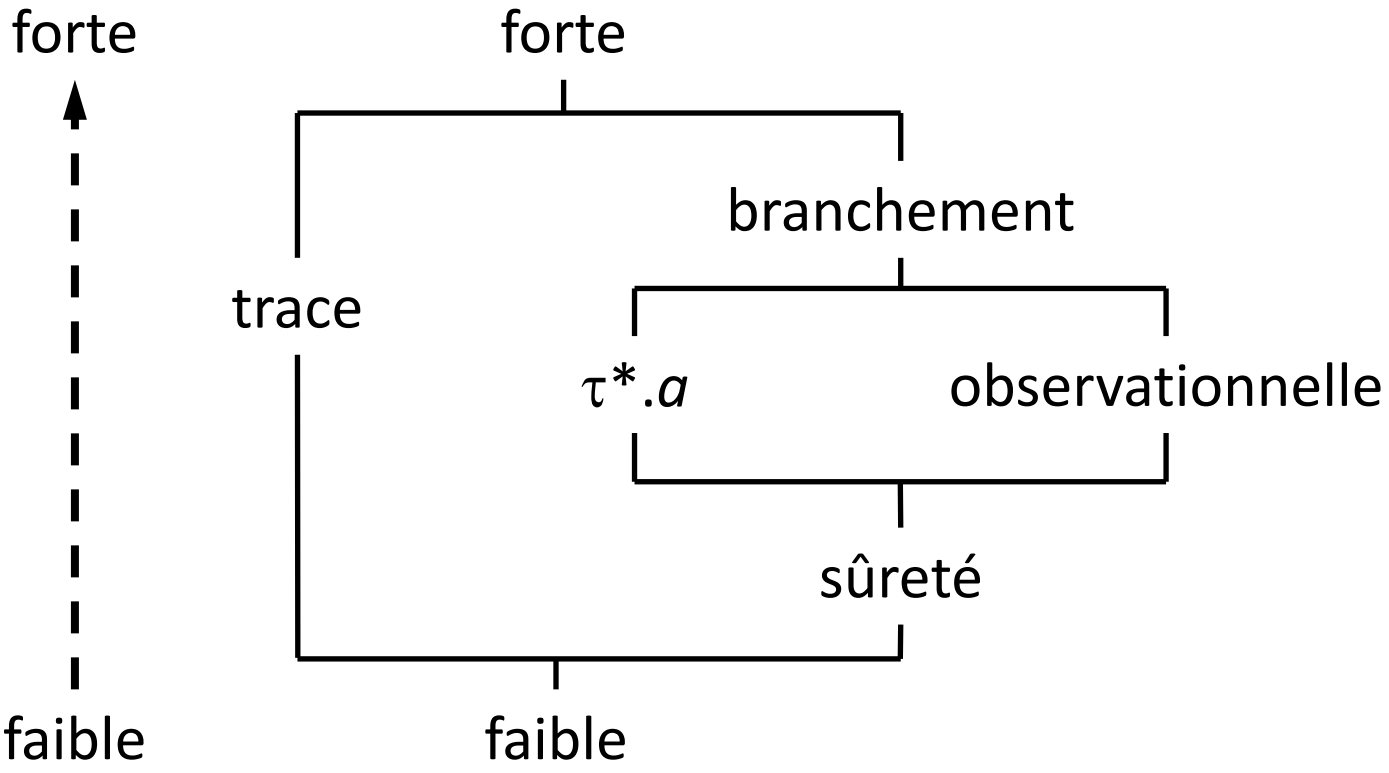
Absence de deadlock



- Blocage créé par le protocole
  - LTS : possibilité d'atteindre action ou blocage
  - propriété MCL : `[true*] ( (<i*> [true] false) implies [true* . 'ACTION'] false )`

Equivalence spéc./implém.

- Comparaison comportement : peut-on les distinguer ?
- Relations d'équivalences disponibles dans CADP :



Vérification de protocoles

- Automatisation de la méthode
  - génération du modèle implém., puis 3 vérifications
- Suite de tests
- **[Evrard-Lang-13]** : étude 3 protocoles
  - [Sisto-Ciminiera-Valenzano-91]
  - [Parrow-Sjödin-92]
  - [Parrow-Sjödin-96] : deadlock possible, correction
- Plus tard : étude α-*core* [Perez-Corchuelo-Toro-04]
  - retrouve deadlock [Katz-Peled-10]

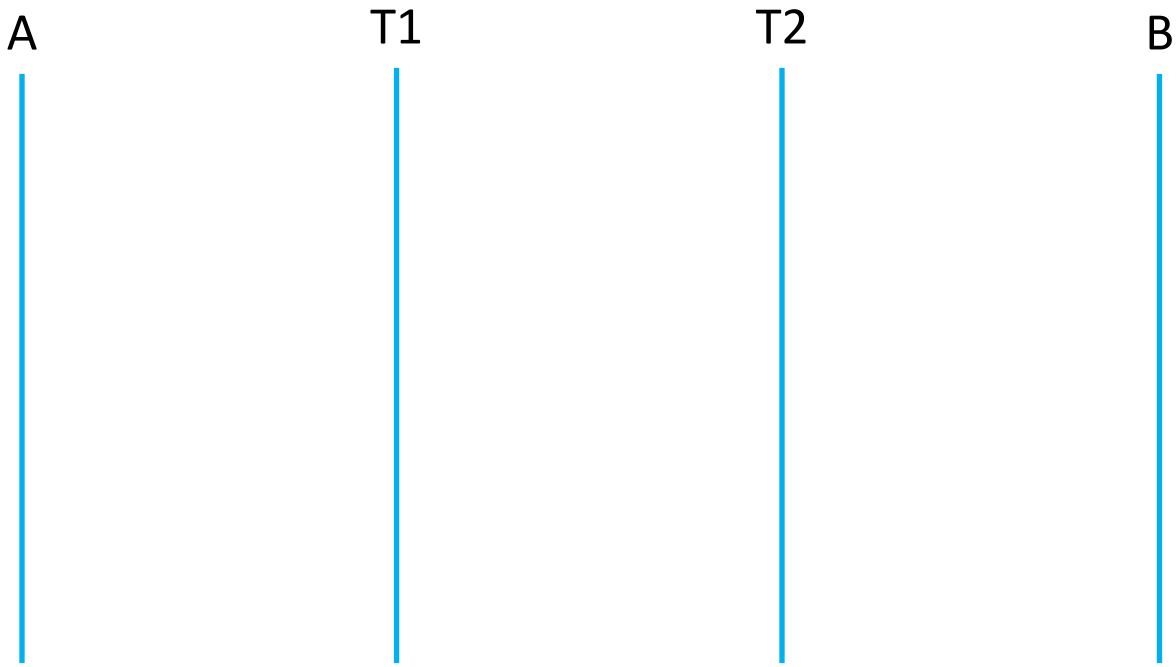
Protocole de rendez-vous

Protocole de rendez-vous

- Base : [Parrow-Sjödin-96] corrigé
  - consomme peu de messages
  - extensible pour LNT (n-parmi-m)
- Processus auxiliaires : portes
- Exclusion mutuelle des conflits
  - verrouillage des tâches par les portes
  - tâche : ressource partagée entre portes
  - ordre sur les tâches pour éviter interblocages

Parrow-Sjödin 96

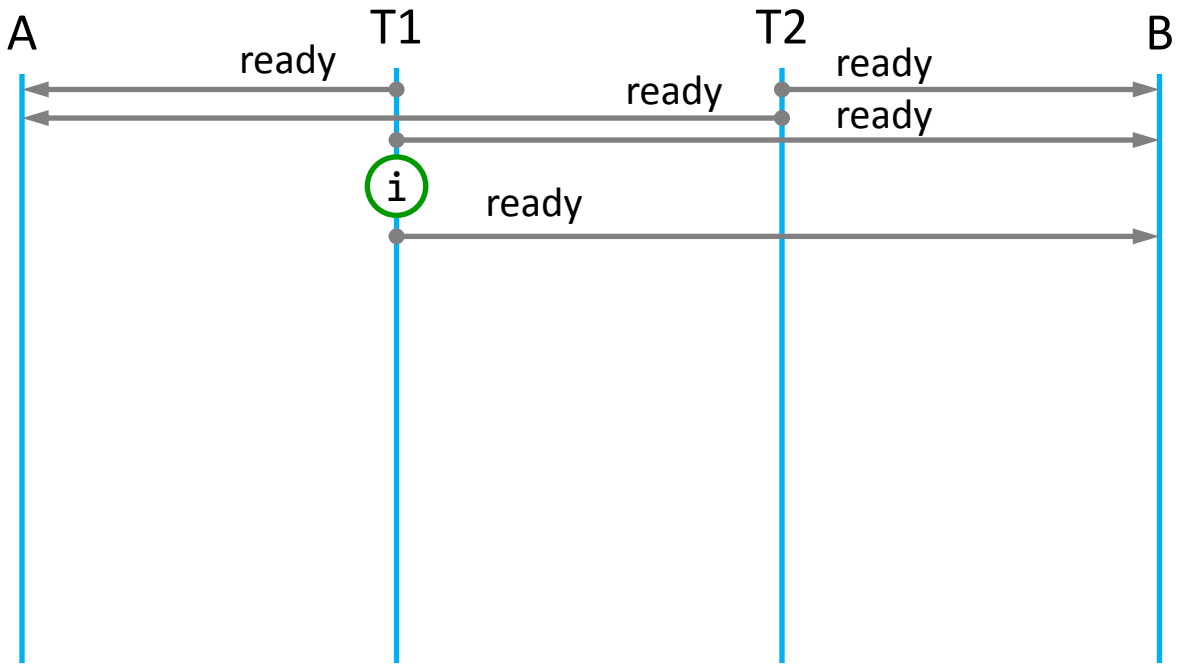
```
process T1 [A,B] is
   select
      A
   [] B
   [] i ; B
   end select
end process

process T2 [A,B] is
   select
      A
   [] B
   end select
end process

par A, B in
   T1 [A,B]
|| T2 [A,B]
end par
```



- Annonces : ready
- Verrouillage : lock
- Résultats : commit / abort

Parrow-Sjödin 96

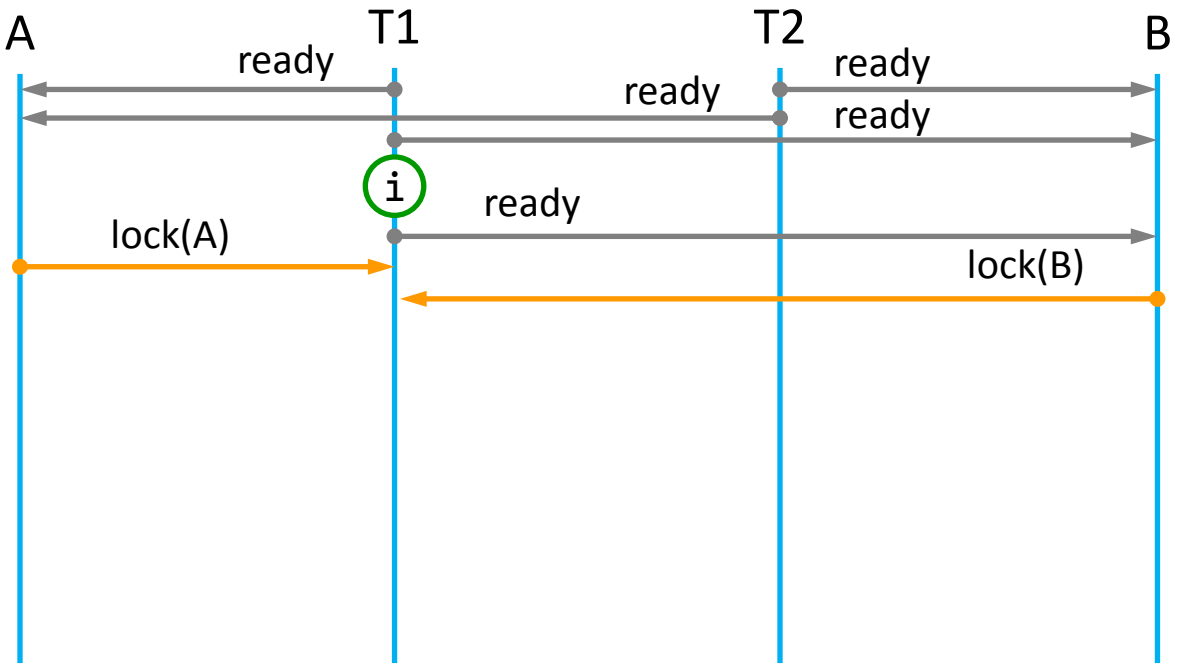
```
process T1 [A,B] is
   select
      A
   [] B
   [] i ; B
   end select
end process

process T2 [A,B] is
   select
      A
   [] B
   end select
end process

par A, B in
   T1 [A,B]
|| T2 [A,B]
end par
```



- Annonces : ready
- Verrouillage : lock
- Résultats : commit / abort

Parrow-Sjödin 96

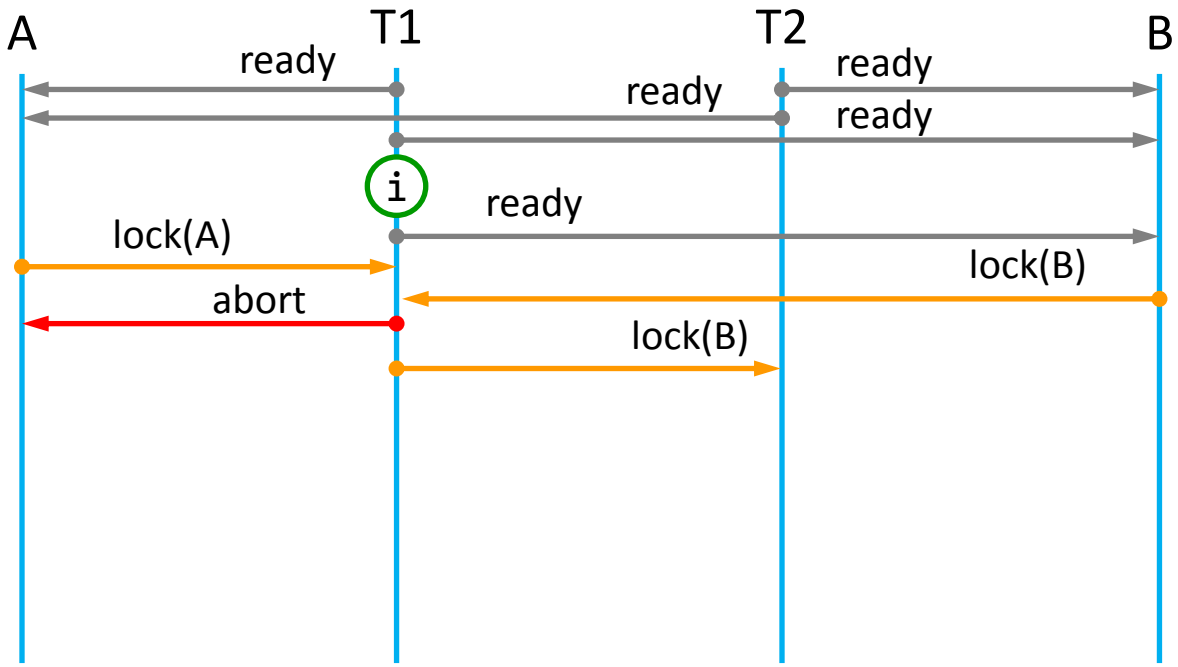
```
process T1 [A,B] is
   select
      A
   [] B
   [] i ; B
   end select
end process

process T2 [A,B] is
   select
      A
   [] B
   end select
end process

par A, B in
   T1 [A,B]
|| T2 [A,B]
end par
```



- Annonces : ready
- Verrouillage : lock
- Résultats : commit / abort

Parrow-Sjödin 96

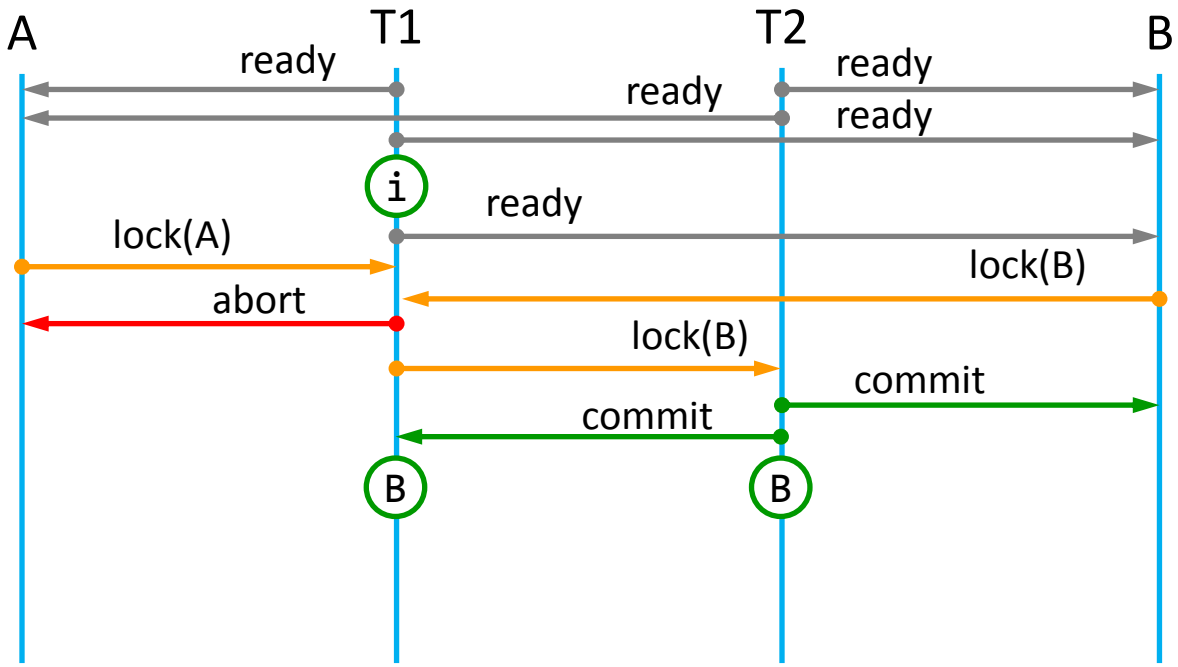
```
process T1 [A,B] is
   select
      A
   [] B
   [] i ; B
   end select
end process

process T2 [A,B] is
   select
      A
   [] B
   end select
end process

par A, B in
   T1 [A,B]
|| T2 [A,B]
end par
```



- Annonces : ready
- Verrouillage : lock
- Résultats : commit / abort

Parrow-Sjödin 96

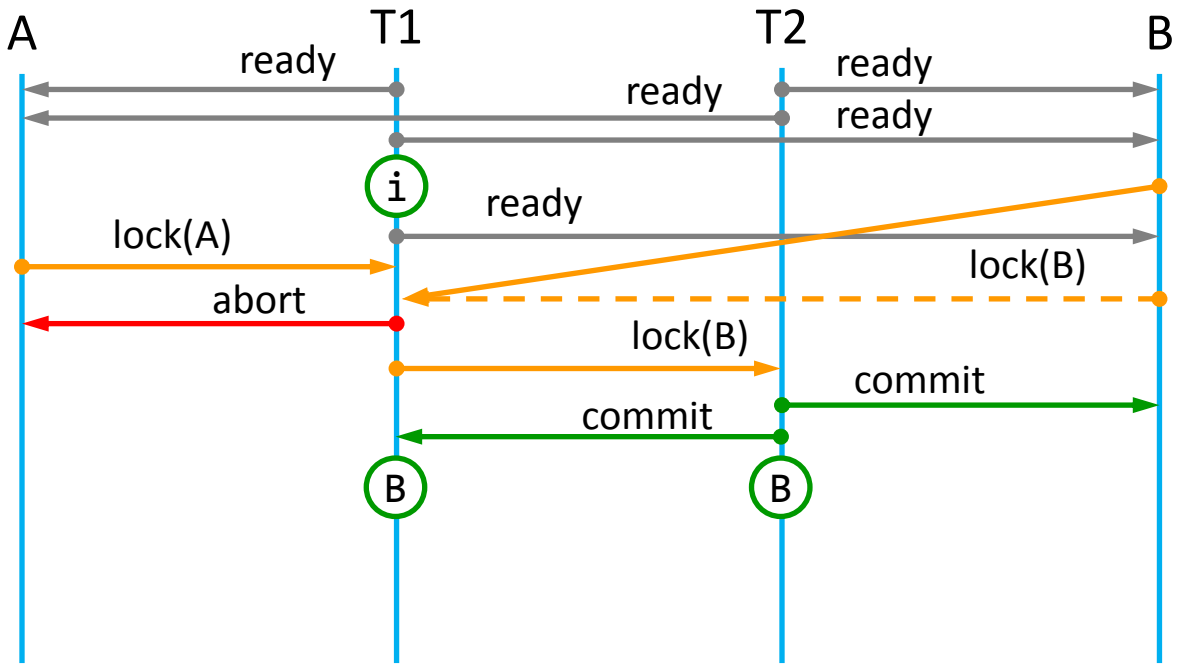
```
process T1 [A,B] is
   select
      A
   [] B
   [] i ; B
   end select
end process

process T2 [A,B] is
   select
      A
   [] B
   end select
end process

par A, B in
   T1 [A,B]
|| T2 [A,B]
end par
```



- Annonces : ready
- Verrouillage : lock
- Résultats : commit / abort

Parrow-Sjödin 96

```
process T1 [A,B] is
   select
      A
   [] B
   [] i ; B
   end select
end process

process T2 [A,B] is
   select
      A
   [] B
   end select
end process

par A, B in
   T1 [A,B]
|| T2 [A,B]
end par
```



- Annonces : ready
- Verrouillage : lock
- Résultats : commit / abort
- Rendez-vous en avance de phase

Contributions protocole

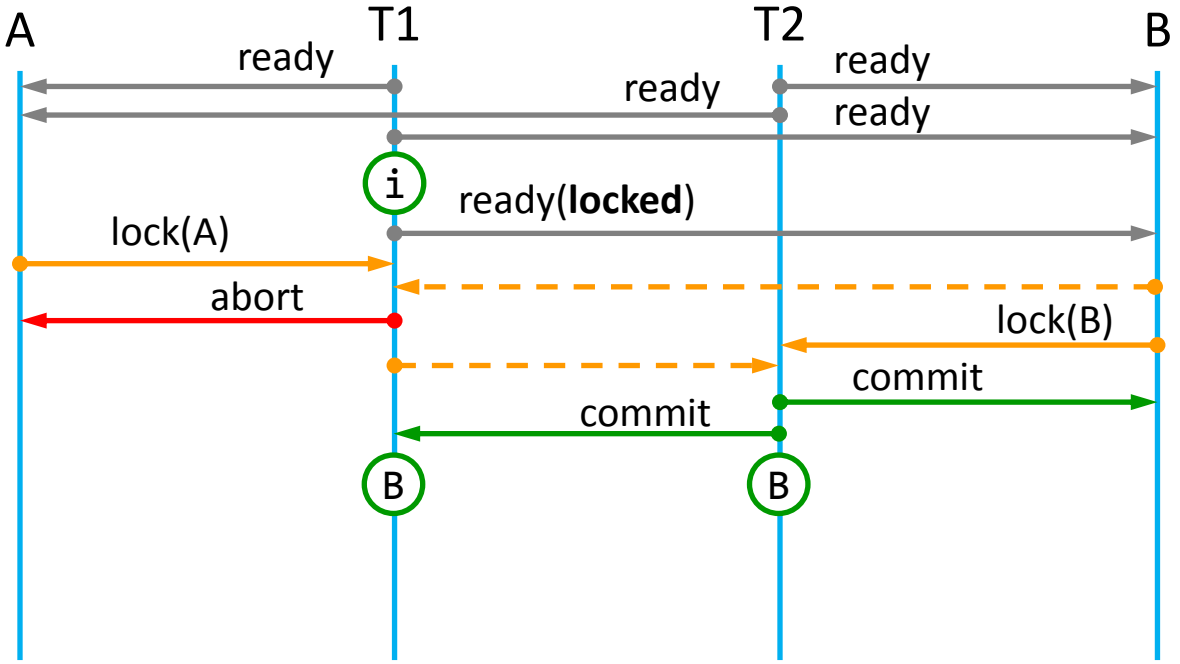
- Itérations sous contrôle de vérification formelle
- Contributions
  - simplifications
    - fusion tâches et managers
    - réduction du nombre de types de messages : de 9 à 4
  - communications asynchrones
  - diffusion en parallèle des résultats (commit/abort)
  - plusieurs vecteurs de synchronisation par porte
  - gestion actions internes
  - gestion des offres (non orthogonal)
  - confirmation par la porte (fonctions crochets)
  - auto-verrouillage
  - mécanisme de purge

Contributions protocole

- Itérations sous contrôle de vérification formelle
- Contributions
  - simplifications
    - fusion tâches et managers
    - réduction du nombre de types de messages : de 9 à 4
  - communications asynchrones
  - diffusion en parallèle des résultats (commit/abort)
  - plusieurs vecteurs de synchronisation par porte
  - gestion actions internes
  - gestion des offres (non orthogonal)
  - confirmation par la porte (fonctions crochets)
  - **auto-verrouillage**
  - **mécanisme de purge**

Auto-verrouillage

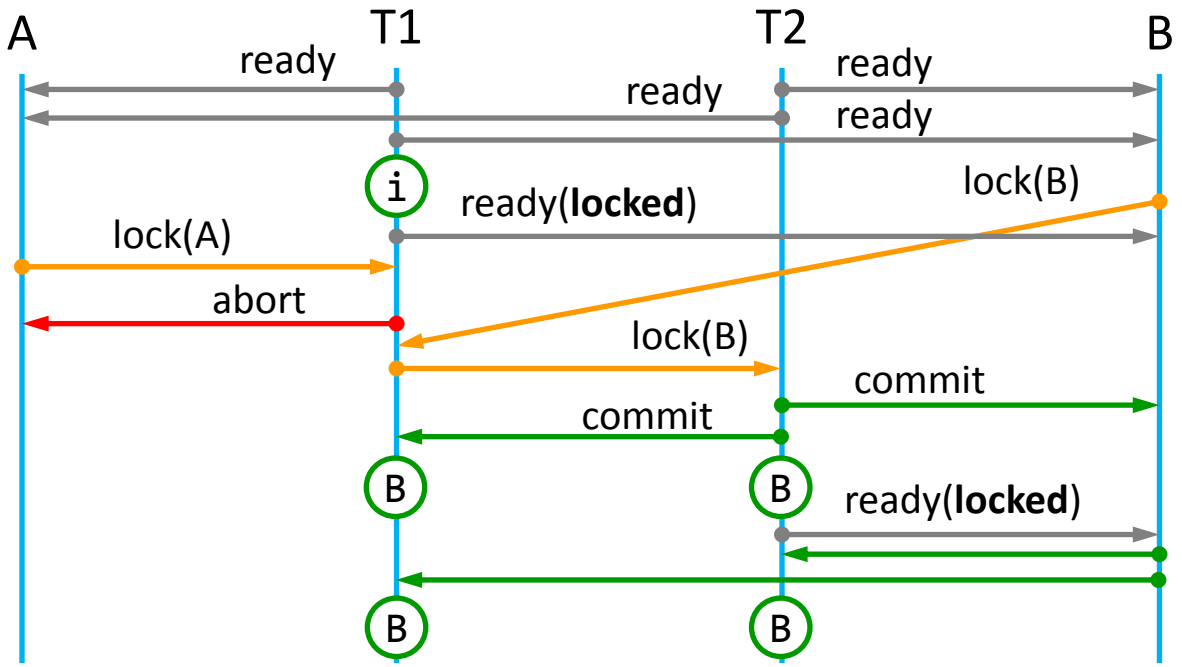
```
process T1 [A,B] is
   select
      A
   [] B
   [] i ; B
   end select
end process

process T2 [A,B] is
   select
      A
   [] B
   end select
end process

par A, B in
   T1 [A,B]
|| T2 [A,B]
end par
```



- Verrouillage : exclusion mutuelle des conflits
- Tâche prête sur 1 seule action : pas de conflit
- Auto-verrouillage : ready(**locked**)
- La négocation nécessite moins de messages
- Mécanisme similaire dans *α-core* [PCT-04]

Auto-verrouillage : problème

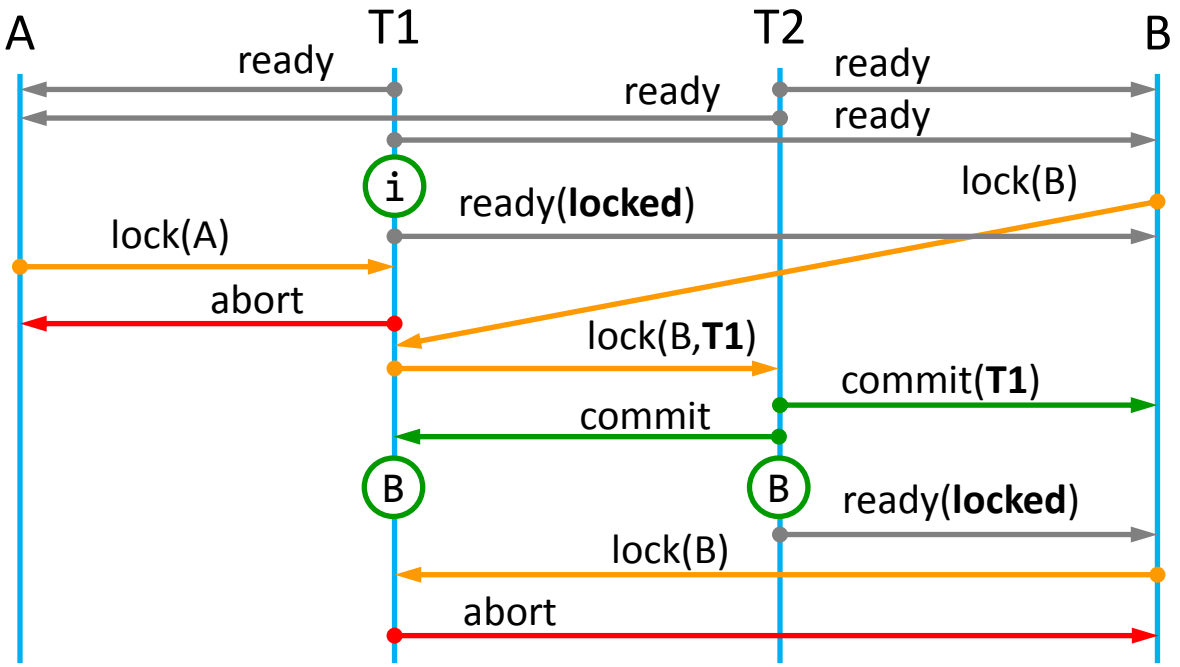
```
process T1 [A,B] is
   select
      A
   [] B
   [] i ; B
   end select
end process

process T2 [A,B] is
   select
      A
   [] B ; B
   end select
end process

par A, B in
   T1 [A,B]
|| T2 [A,B]
end par
```



- B considère T1 et T2 auto-verrouillée
- 2ème action sur B **invalide** pour T1

Solution : purge

```
process T1 [A,B] is
   select
      A
   [] B
   [] i ; B
   end select
end process

process T2 [A,B] is
   select
      A
   [] B ; B
   end select
end process

par A, B in
   T1 [A,B]
|| T2 [A,B]
end par
```



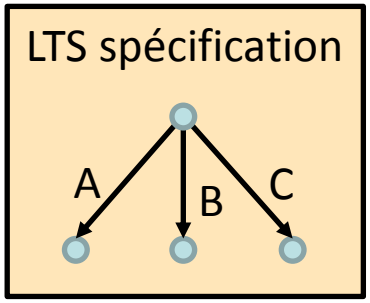
- T1 reçoit lock alors qu'elle est auto-verrouillée
- T1 signale qu'il faut purger son ready(locked)
- B considère seulement T2 auto-verrouillée
- Pas d'action invalide pour T1

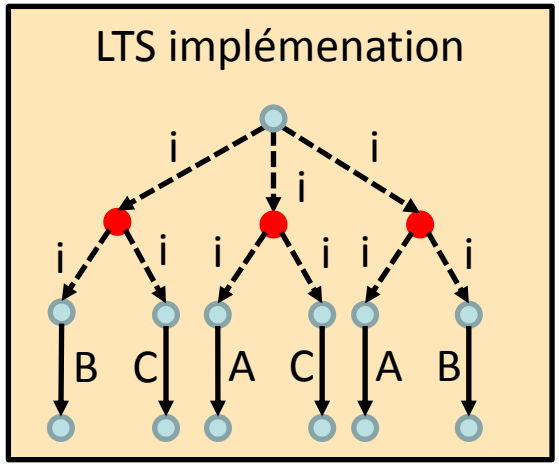
Complexité du protocole

- Rendez-vous : n tâches, dont k auto-verr. ($k <= n$)
- Nombre total de messages
  - Parrow-Sjödin : $3n$
  - α-*core* : $4n - 2k + x$
  - DLC : $3n - k$
- Plus longue séquence de messages
  - Parrow-Sjödin : $2n$
  - α-*core* : $4 + 2(n - k)$
  - DLC : $2 + n - k$
- Purge : permet auto-verrouillage, sans ajout de messages

Vérification du protocole

- Absence de livelock et de deadlock
- Equivalence :
  - relation de sûreté (*safety*)
  - choix d'action non-atomique

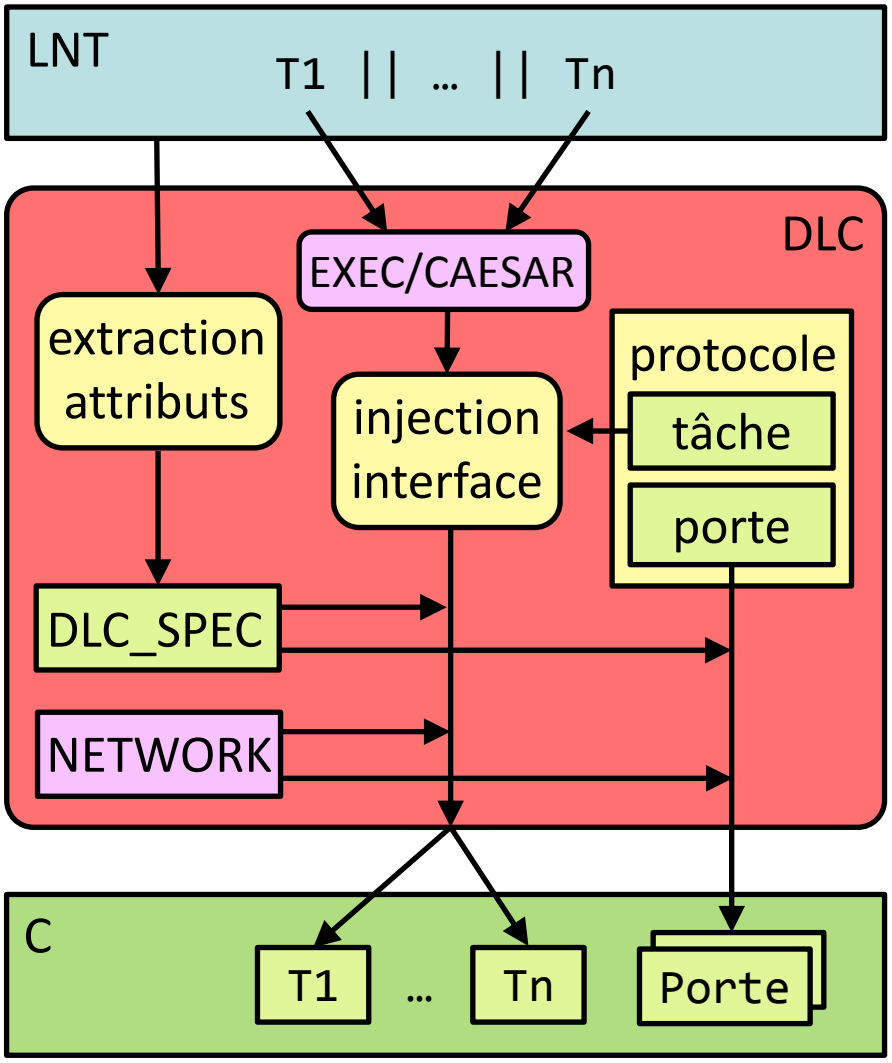




- Relation coupled simulation [Parrow-Sjödin-92]
  - pas implémentée dans CADP

Distributed LNT Compiler

Architecture DLC



- Script principal
  - 1500 loc. shell
- Protocole de rendez-vous
  - modules génériques
  - 2200 loc. C
- Génération de code
  - caractéristiques de la spécification : DLC_SPEC
  - interface tâche / protocole dans code émit par EXEC/CAESAR
  - gestion offres
  - code généré : taille variable

Limitations DLC

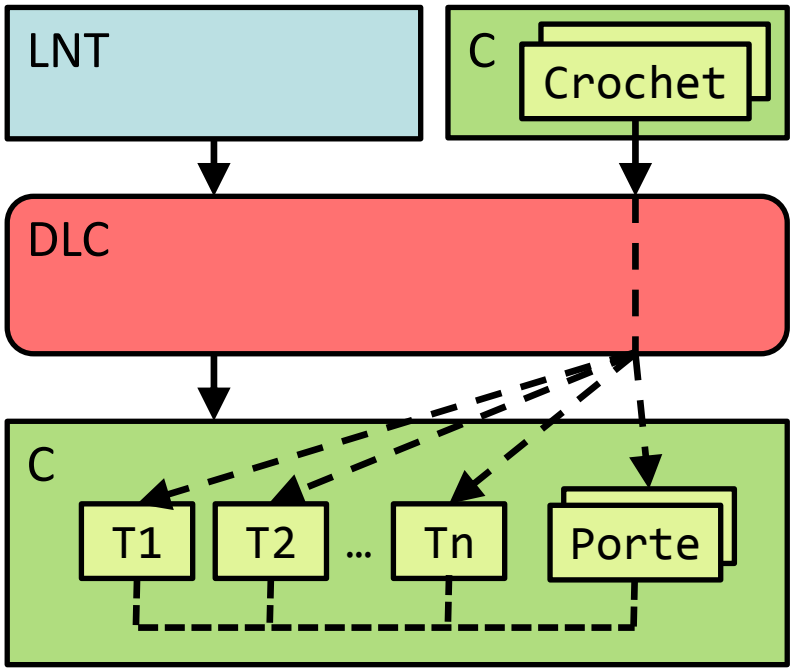
- Données rendez-vous : types complexes
  - liste, tableau, … non supportés
  - limitation technique : sérialisation en chaîne d'octets
  - modification compilateur CAESAR.ADT
- Action gardée
  - `A(?n) where n > 42`
  - EXEC/CAESAR : pas d'accès aux gardes
- Création dynamique de tâche
  - modification EXEC/CAESAR
  - protocole : mise à jour des vecteurs de synchronisation

Fonctions crochets

Interactions avec l'environnement

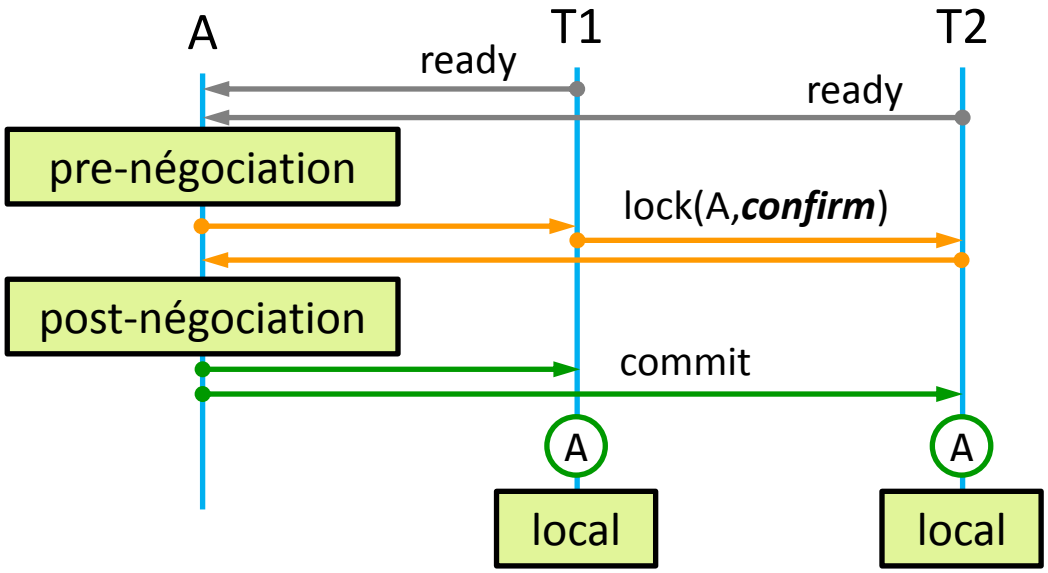
- Réalisation d'effets de bords
  - input / output
  - commande d'actionneurs
- Contrôle de l'exécution
  - influence sur le choix des actions réalisées
  - autorisation d'action après un certain délai
- Via fonctions LNT ?
  - possibilité de définir des fonctions en C …
  - … hypothèse : fonctions pures, sans effets de bords

Fonctions crochets



- Fonctions C optionnelles, définies par l'utilisateur
  - embarquées au sein de l'implémentation générée
  - déclenchement lié à une action sur porte

Trois types de crochets



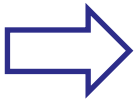
- pre-négociation : évite négociation inutile
- post-négociation : décide réalisation action
  - confirmation par la porte
- local : effet de bord au niveau tâche

Échange de données (1)

- Crochet : accès aux offres
- Passage donnée : système → environnement
  - offre en mode envoi
  - lecture valeur de l'offre

LNT

```
process … is
   …
   n := 42;
   A(n);
   …
end process
```



C – Crochet porte A

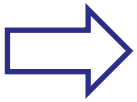
```
bool DLC_HOOK_PRE_NEGOTIATION (action *a)
{
   DLC_VALUE i;
   …
   i = a->offers[0].value;
   …
}
```

Échange de données (2)

- Passage donnée : environnement → système
  - offre mode réception
  - affectation valeur, mise à jour mode

LNT

```
process … is
    …
    A(?n);
    …
end process
```



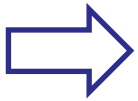
C – Crochet porte A

```
bool DLC_HOOK_PRE_NEGOTIATION (action *a)
{
    …
    a->offers[0].value = foobar();
    a->offers[0].mode  = DLC_MODE_SEND;
    …
}
```

Action retardée

LNT

```
process … is
    …
    SLEEP(n);
    …
end process
```



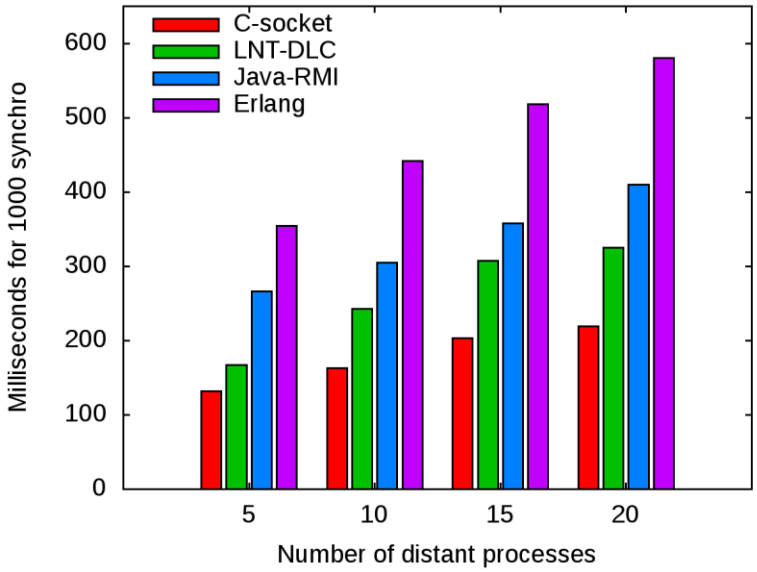
C – Crochet porte SLEEP

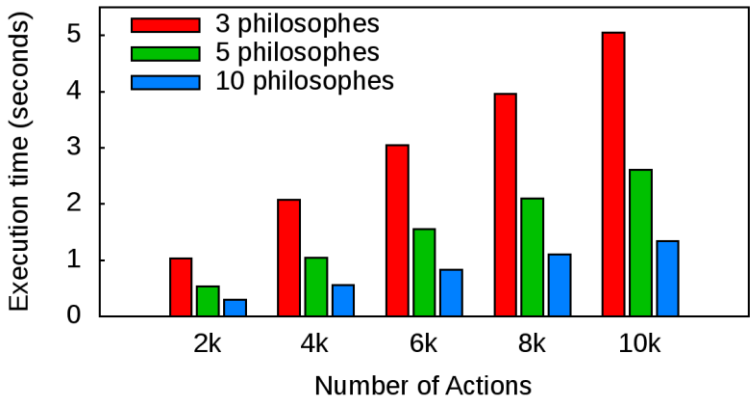
```
time_t debut = 0;

bool DLC_HOOK_PRE_NEGOTIATION (action *a)
{
    if (debut == 0)
        { debut = time(); }
    time_t attente = time() - debut;
    return (attente >= a->offers[0].value)
}

bool DLC_HOOK_POST_NEGOTIATION (action *a)
{
    debut = 0;
    return TRUE;
}
```

Expérimentations

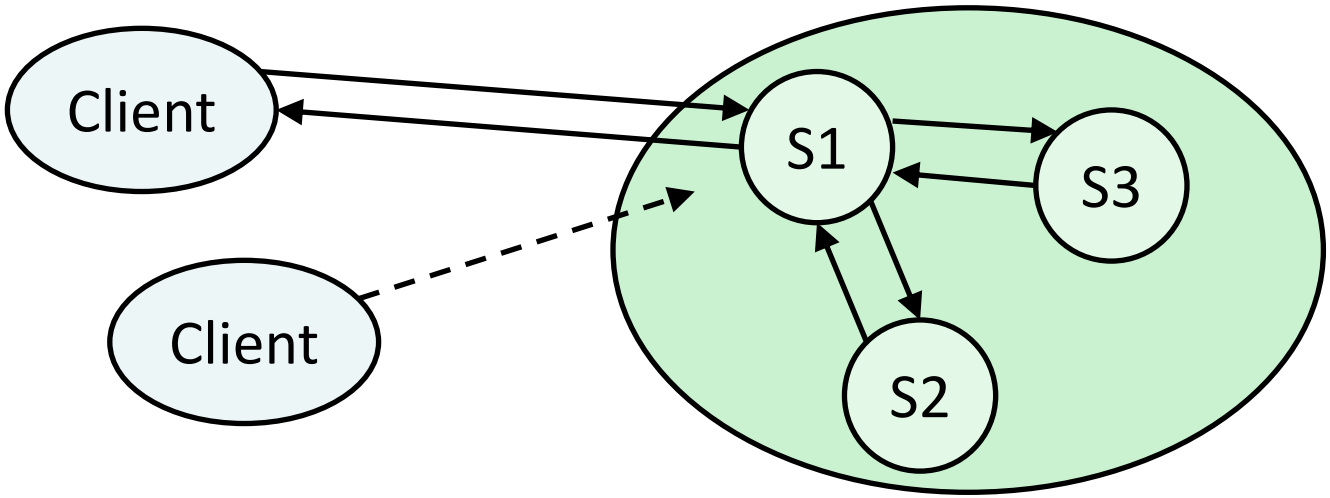
Perfomances Rendez-vous

- Barrière de synchronisation
  - cas auto-verrouillage
  - 1000 rdv 10 proc. : 250 ms (0,25 ms par rendez-vous)
  - plus rapide que Java/Erlang
- Philosophes
  - conflits possibles
  - en séquence : 2000+ rdv/sec
  - rendez-vous indépendants réalisés en parallèle





Cas d'étude : Raft



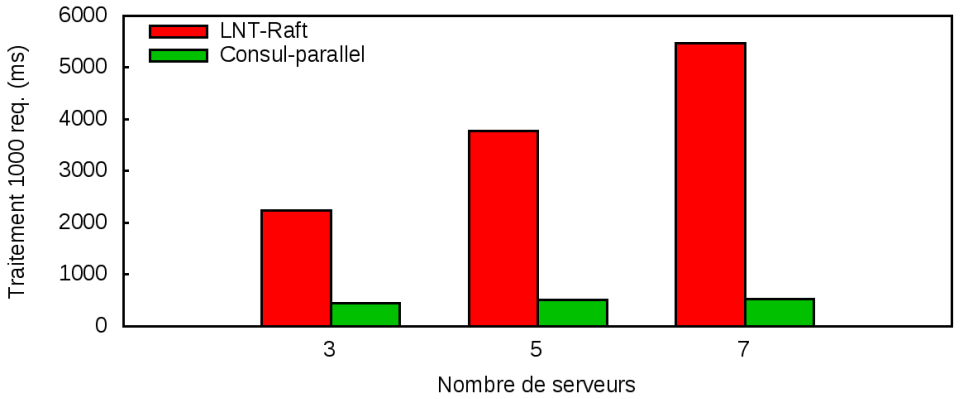
- Raft consensus [Ongaro-Ousterhout-14]
  - réplication de requêtes clients
  - tolérant aux fautes
  - leader unique (élection, pulsations)
- Outils indus. : etcd (CoreOS), Consul (Hashicorp)

LNT Raft

- Modèle LNT Raft
  - découverte bogue dans spécification TLA+ des auteurs
  - 400 loc. LNT (Consul : 4000 loc. Go)
- Crochets
  - délais : élection leader, pulsations
  - client : programme externe, interface TCP
  - 300 loc. C (principal : serveur TCP)
  - bon cas d'étude pour élaboration crochets
- Code généré
  - 7400 loc. C par serveur

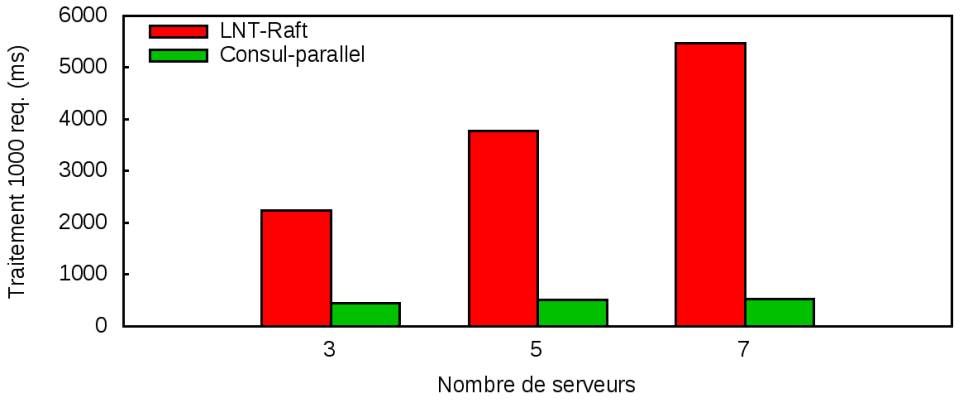
LNT Raft – Performances

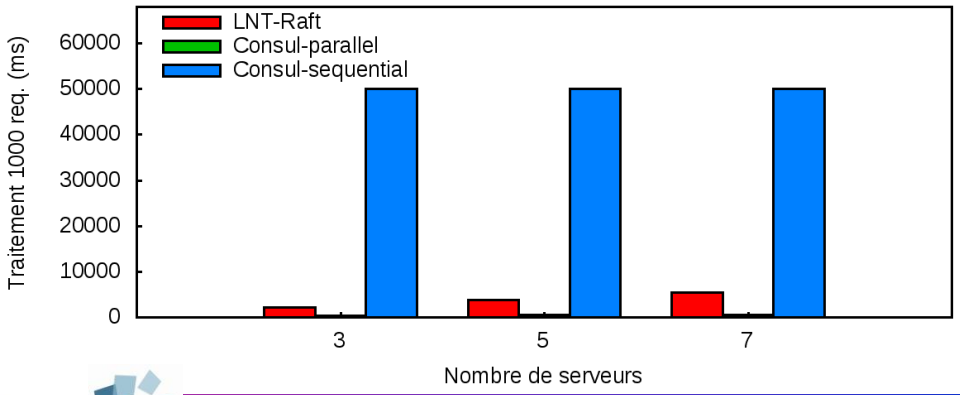
- LNT Raft
  - 1000 req. / 7 srv. : 14000 rdv en 5,5s. (5ms/req et 0,4ms/rdv)
  - leader unique : rendez-vous en séquence
- LNT Raft versus Consul
  - débit : Consul plus rapide
  - Nb serv. : pas d'impact…



LNT Raft – Performances

- LNT Raft
  - 1000 req. / 7 srv. : 14000 rdv en 5,5s. (5ms/req et 0,4ms/rdv)
  - leader unique : rendez-vous en séquence
- LNT Raft versus Consul
  - débit : Consul plus rapide
  - Nb serv. : pas d'impact…
- …Consul : optimisation
  - groupement requêtes
  - réplication toutes les 50ms
  - latence : Consul plus lent
- LNT : optim. nécessite liste





Conclusion

Principales contributions

- Méthode de vérification formelle de protocoles
  - détection de deadlock dans [Parrow-Sjödin-96]
- Conception d'un protocole efficace, vérifié, générique
- Mise en oeuvre dans l'outil DLC
- Interactions avec l'environnement
  - fonctions crochets
- Expérimentations
  - bonnes performances pour le prototypage rapide
- Extension du cadre de développement formel pour systèmes distribués

Publications

- *Formal Verification of Distributed Branching Multiway Synchronisation Protocols,* FORTE/FMOODS 2013
  - http://hal.inria.fr/hal-00818788
- *Automatic Distributed Code Generation from Formal Models of Asynchronous Processes,* PDP–4PAD 2015
  - http://hal.inria.fr/hal-01086522
  - extension pour JLAMP (*Journal of Logic and Algebraic Methods in Programming*) en cours

Travaux futurs

- Nouveaux cas d'études de systèmes distribués
- DLC : lever les limitations actuelles
  - types complexes, gardes, création dynamique tâche
- Protocole de rendez-vous
  - optimisation cas spécifiques (ex: rdv à 2)
  - preuve du protocole (coupled simulation)
- Actions retardées
  - LNT : ajout notion temps [Sighireanu-99]
- Vérification à l'exécution (co-simulation)
- Amorçage DLC à partir de LNT (*bootstrapping*)

Remerciements

Questions

[Parrow-Sjodin-96] Async. Comm.

- *Designing a Multiway Synchronisation Protocol*

2 A Model of Multiway Synchronization

We assume that there are n processes, in the following called *client processes*, communicating over a communication network providing asynchronous point-to-point connections (see Fig. 1). Thus messages will arrive after an unknown and variable delay, and the network never loses messages.

We further assume that there are several multiway synchronization *actions*, denoted a, b

- Modèle de communication asynchrone
  - passage de message : rdv synchrone
  - buffer interne au processus via liste
- [PS-96] réaction dès la réception : msg synchrones

Vérification de Raft (1)

- Modèle paramétré
  - nb serveurs
  - nb requêtes
  - term : nb chgt de leader (pannes, timeout elections)
  - buffers : taille (mais aussi : maintient ordre, perte)
- Distributor sur Grid5000
  - 3 srv, 3 req, 2 term : 114 M états, 643 M trans.
    - reduit pour State Machine Safety : 494 états, 1332 trans.
  - 3 srv, 2 req, 3 term : 1,5 G états, 9 G transitions
- Modèle trop détaillé ? nécessité pour implem ?

Vérification de Raft (2)

- Approche compositionnelle
  - interface serveurs
  - pannes / redémarrage des autres serveurs : difficile de borner un seul serveur
- Outils de vérifications distribués
  - certains espace d'états peuvent être générés mais non réduits : pas asser de RAM sur une machine unique

Calcul complexité

- Rendez-vous : n tâches, dont k auto-verr. ($k <= n$)
- Parrow-Sjödin
  - n ready + n lock + n commit = $3n$
- α-*core*
  - phase re-synchronisation : x messages
  - n ready + $2(n - k)$ locks + n commit + x = $4n - 2k + x$
- DLC
  - n ready + $(n - k)$ lock + n commit = $3n - k$
- Avance de phase : réduction du coût
- Purge : pas d'ajout de message

Complexité : plus longue séquence

- Rendez-vous : *n* tâches, dont *k* auto-verr. (*k* <= *n*)
- Parrow-Sjödin
  - 1 ready + *n* lock + (*n* – 1) commit = 2*n*
- *α-core*
  - si n = k : 1 ready + 1 commit = 2
  - si n > k : 1 ready + 2(n - k) lock + 1 refuse + 1 ack = 4 + 2(n-k)
- DLC
  - 1 ready + (*n* - *k*) lock + 1 commit = 2 + *n* - *k*

DLC bootstrapp

- OPEN/CAESAR : pas d'offre en réception, énumération

- A(?b) → liste action possible :
  - A (true)
  - A (false)
- EXEC/CAESAR :
  - A (offre : bool, mode réception)

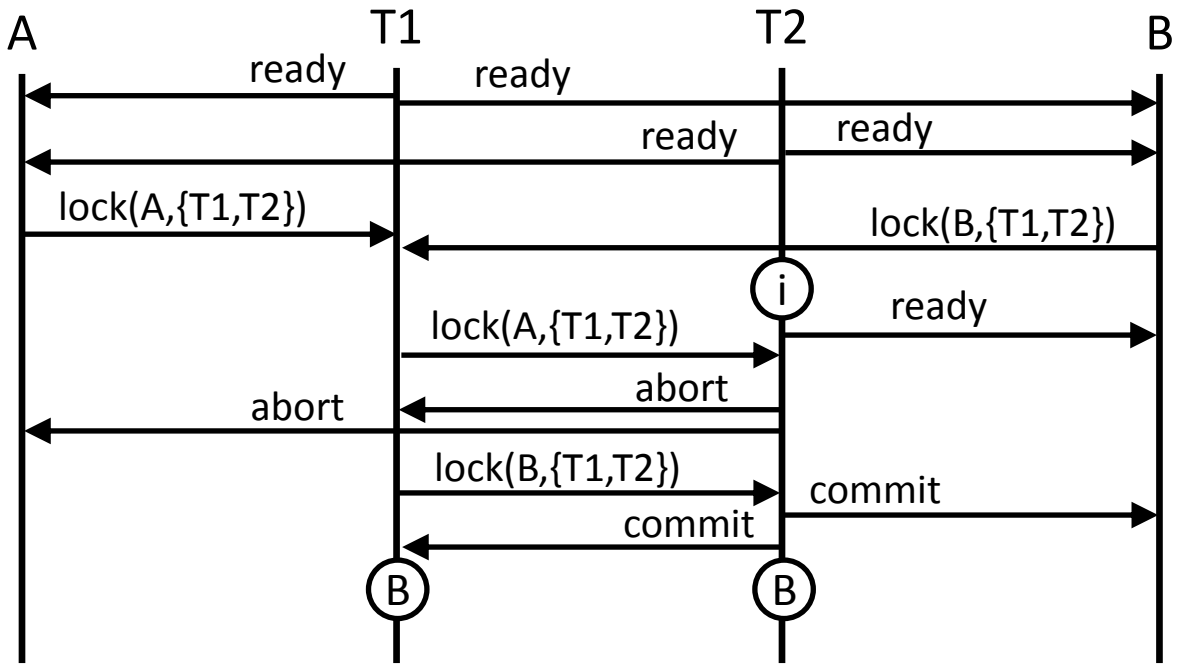
backup

Bilan Crochets

- Permet interaction avec environnement
  - échange de données
- Permet contrôle actions du système
  - interdit des actions possibles
  - actions retardées

Base : [Parrow-Sjödin-96]

```
par A, B in
   T1 [A,B]
|| T2 [A,B]
end par

process T1 [A,B] is
   select
      A
   [] B
   end select
end process

process T2 [A,B] is
   select
      A
   [] B
   [] i; B
   end select
end process
```



- 3 phases : annonces, verrouillage, résultats
- Rendez-vous en avance de phase
  - négociation périmée peut réussir

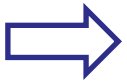
LNT – Rendez-vous

- Rendez-vous en conflit
  - nécessitent la participation de processus communs

```
par A, B in
   select A [] B end select
|| select A [] B end select
end par
```

- Représentation par vecteurs de synchronisation
  - opérateur n-parmi-m

```
par A, B #2 in
   C -> P1[A,B,C]
||      P2[A,B]
|| C -> P3[A,B,C]
end par
```



- A : (P1,P2,P3)
- B : (P1,P2), (P1,P3), (P2,P3)
- C : (P1,P3)

  - conflits : vecteurs avec processus commun

LNT Raft – Bilan

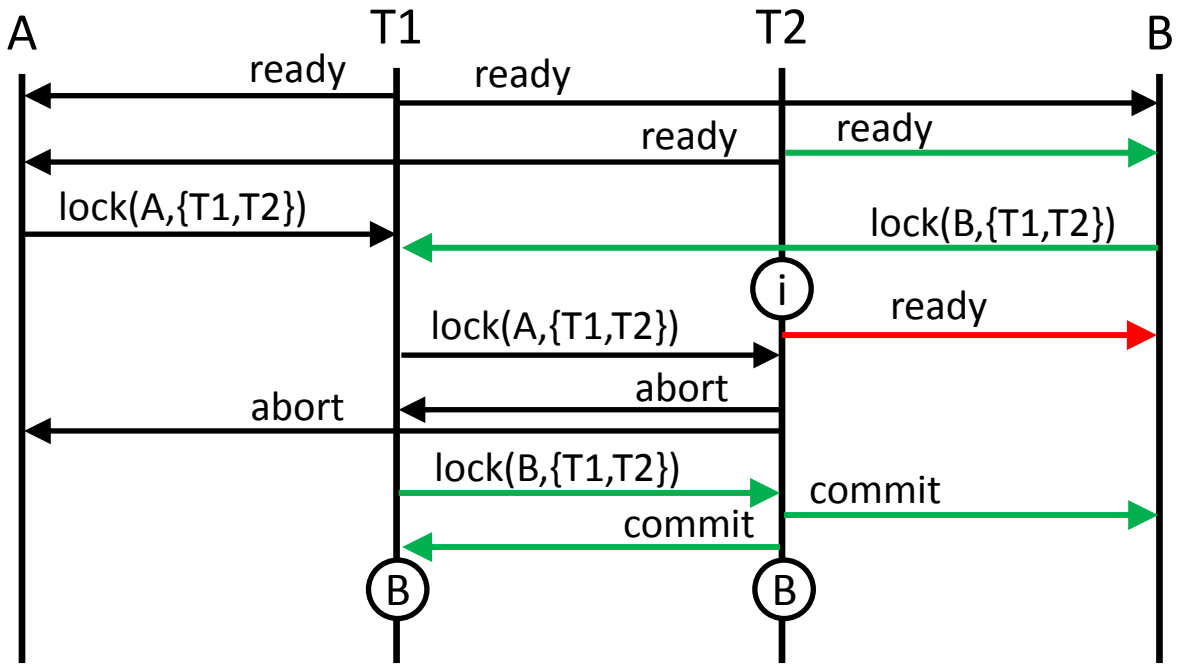
- Algorithme non trivial
  - LNT bon support pour modélisation
- Validation crochets
- Performances
  - 1000+ rdv/sec.
  - pas toujours au niveau implémentation manuelle
  - suffisantes pour prototypage rapide

Utilisation DLC

- Compilateur
  - NETWORK : déploiement intégré
  - sources LNT suffisantes (BIP : partitionnement requis)
- Crochets
  - détection automatique (nom de fichier)
  - création de template
- Rendez-vous avec mélange envoi / réception
  - `par A(42, ?b) || A(?n, TRUE) end par`
- Action interne
  - option progrès maximal

Base : [Parrow-Sjödin-96]

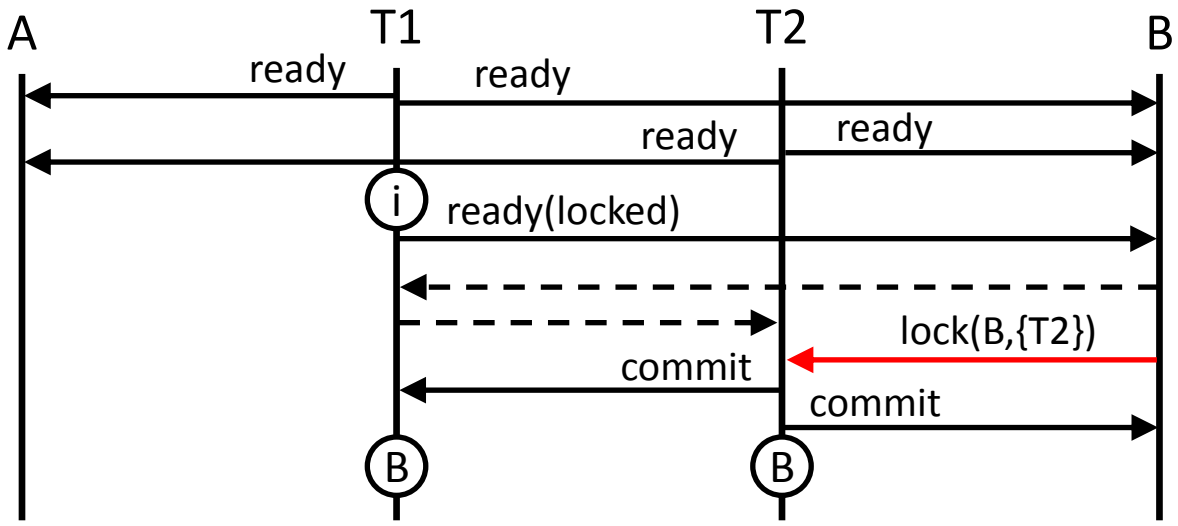
```
par A, B in
   T1 [A,B]
|| T2 [A,B]
end par

process T1 [A,B] is
   select
      A
   [] B
   end select
end process

process T2 [A,B] is
   select
      A
   [] B
   [] i; B
   end select
end process
```



- 3 phases : annonces, verrouillage, résultats
- Rendez-vous en avance de phase
  - négociation périmée peut réussir

Auto-verrouillage

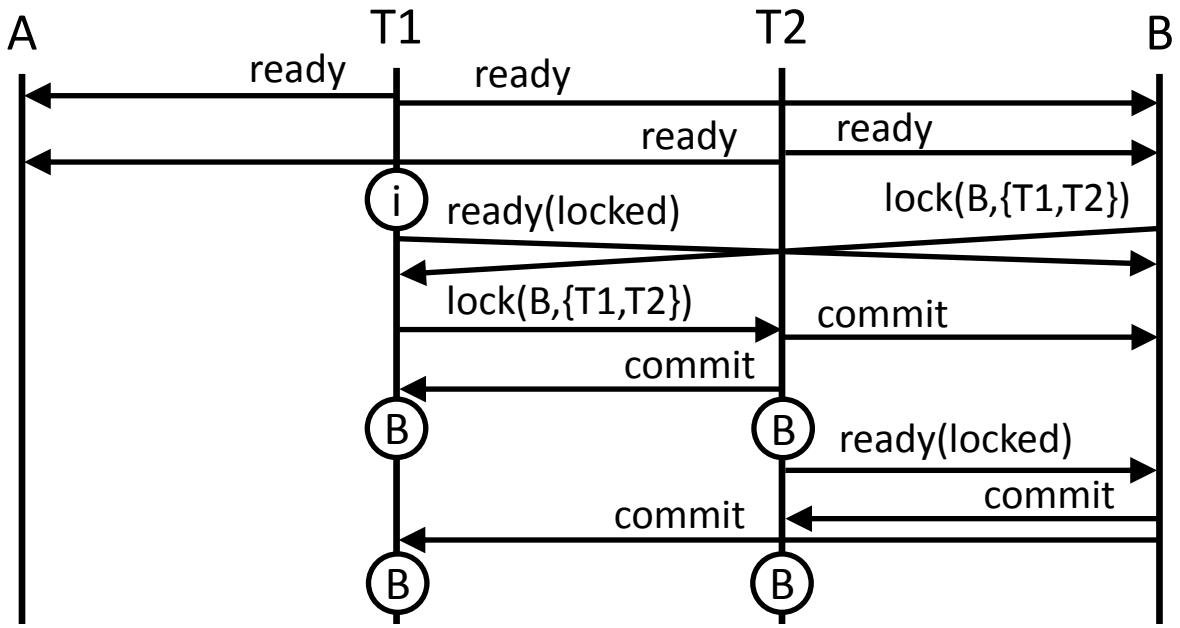
```
par A, B in
   T1 [A,B]
|| T2 [A,B]
end par

process T1 [A,B] is
   select
      A
   [] B
   [] i; B
   end select
end process

process T2 [A,B] is
   select
      A
   [] B
   end select
end process
```



- Tâche prête sur une seule porte
  - pas de conflits
  - auto-verrouillage : ready(locked)
  - pas de lock nécessaire
  - chaîne de verrouillage raccourcie
- Mécanisme similaire dans *α-core*

Probleme

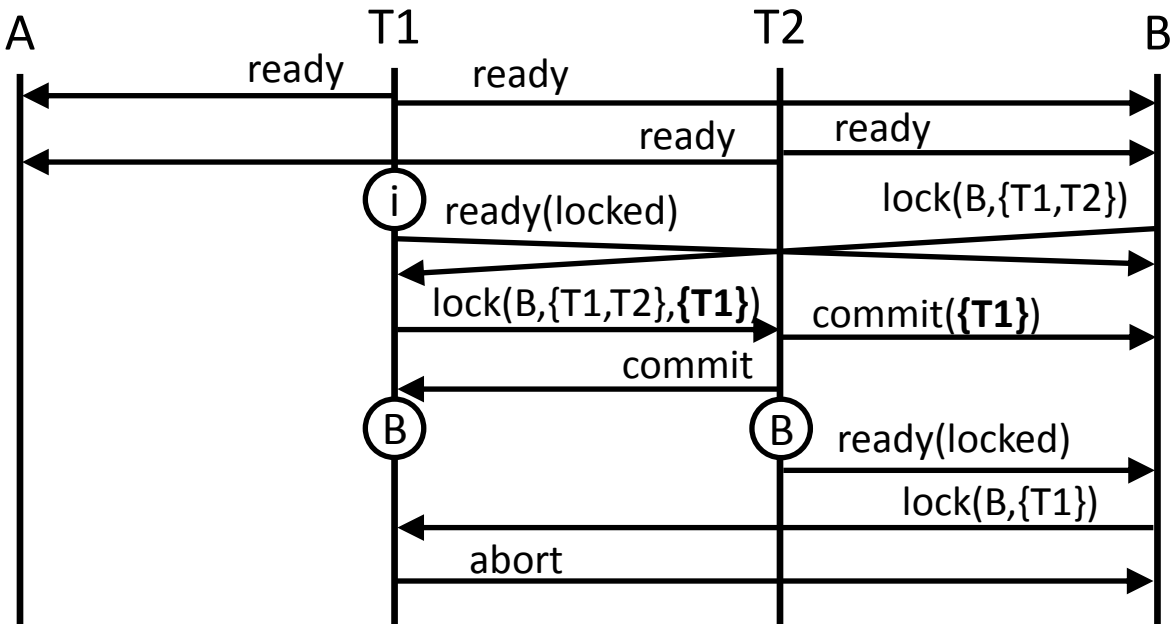
```
par A, B in
   T1 [A,B]
|| T2 [A,B]
end par

process T1 [A,B] is
   select
      A
   [] B
   [] i; B
   end select
end process

process T2 [A,B] is
   select
      A
   [] B; B
   end select
end process
```



- T1 : au plus 1 action sur B
  - B considère T1 et T2 auto-verrouillées
  - action invalide pour T1

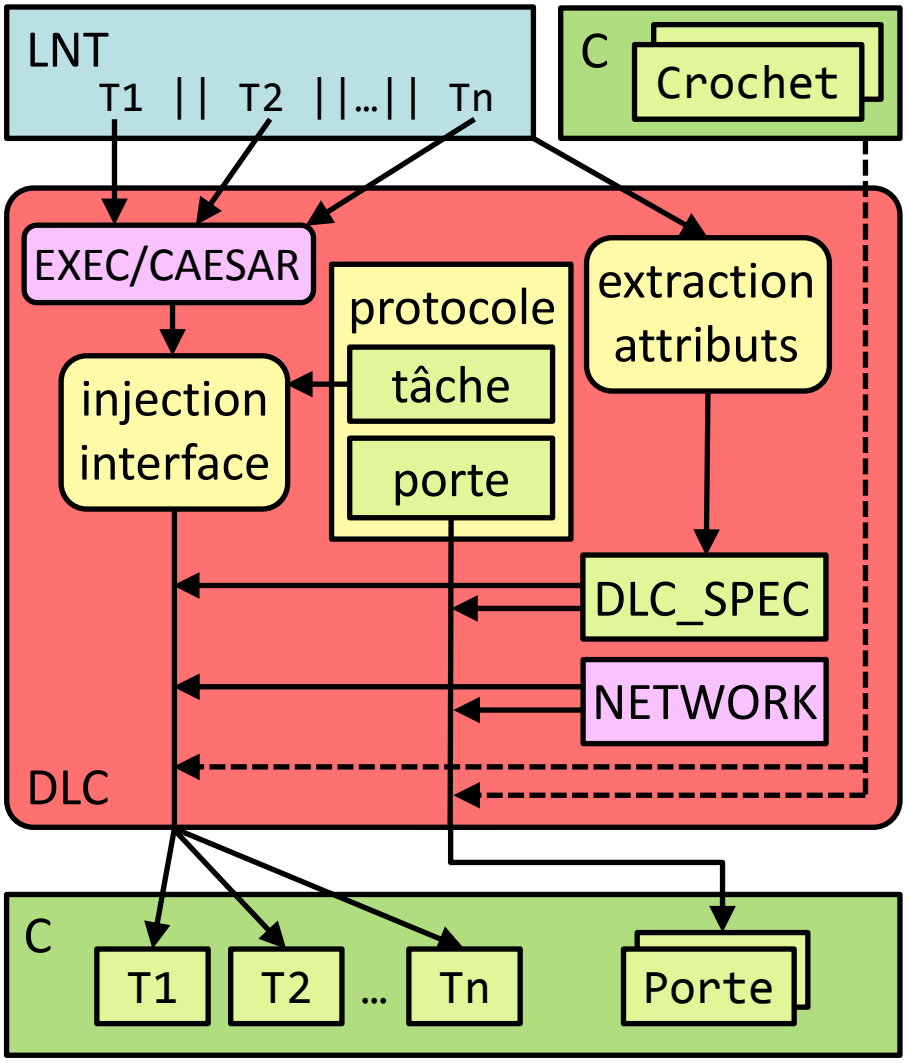
Mécanisme de Purge

```
par A, B in
   T1 [A,B]
|| T2 [A,B]
end par

process T1 [A,B] is
   select
      A
   [] B
   [] i; B
   end select
end process

process T2 [A,B] is
   select
      A
   [] B; B
   end select
end process
```



- T1 : au plus 1 action sur B
  - B considère ~~T1 et~~ T2 auto-verrouillée
  - ~~action invalide pour T1~~
- Purge du message ready(locked)
  - T1 annule son auto-verrouillage

Architecture DLC



- Implémentation outil DLC
  - script compilation : 1500 loc. Shell/AWK
  - protocole (générique) : 2200 loc. C
  - code genere : depend de l'application ; difficilement quantifiable
  - on inclus pas le code genere:
  - DLC_SPEC, sorte de EXEC_CAESAR, etc