

---

# Cours ASPD

## Systèmes Répartis: Problèmes - Concepts - Techniques - Outils

### Telecom Nancy 2A (Apprentissage et IL)

---

Dominique Méry  
Telecom Nancy  
Université de Lorraine

---

**Année universitaire 2025-2026**  
**1<sup>er</sup> février 2026(9:46am)**

# Sommaire

- 1 Problèmes des systèmes répartis
- 2 Exemples de système réparti
  - Problème de l'élection d'un leader
  - Problème du snapshot
- 3 Systèmes répartis : modèles et langages
- 4 Algorithmes répartis



- Des états locaux versus états globaux : blocage, consistance, ...
- Calculs et communications : synchronisation, temporisation, calculs synchrones, calculs asynchrones, architectures réparties, ...
- Fiabilité des calculs et des communications : acheminement des messages, duplication des messages, pertes des messages, sécurisation des données, altération des messages, ...
- Partage de ressources : données sur la vie privée, données pour des éléments de preuve, données financières, ...
- Gestion des fautes et maintien de la qualité de service dans le cadre des réseaux
- Infrastructures hétérogènes

## Scénarios des systèmes répartis

- Un ensemble de machines communiquant via un réseau de communication
- Les communications sont souvent peu fiables et perdent des messages
- Les messages peuvent arriver dans des ordres différents
- Les hypothèses sur les communication ne sont pas toujours faciles à formaliser
- Besoin de modéliser les systèmes et de gérer le temps et les temps.

① Problèmes des systèmes répartis ]

② Exemples de système réparti

Problème de l'élection d'un  
leader

Problème du snapshot

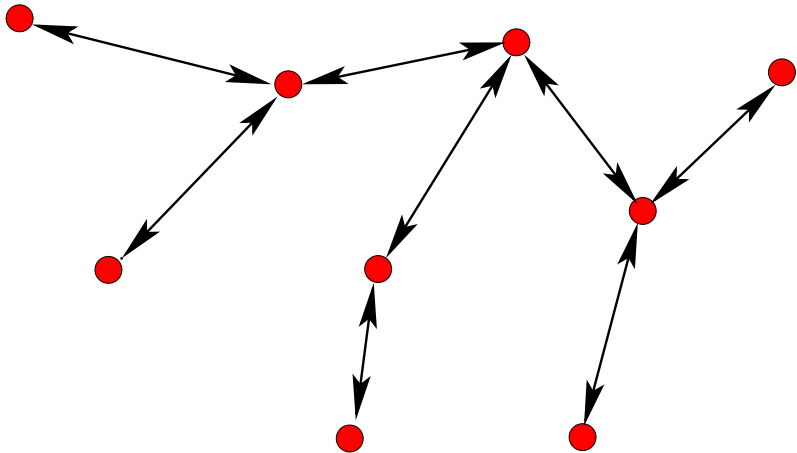
③ Systèmes répartis : modèles et  
langages

④ Algorithmes répartis

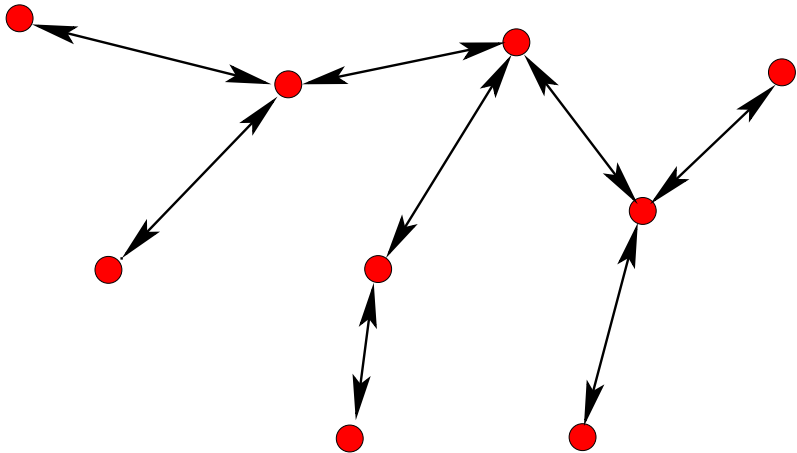
- 1 Problèmes des systèmes répartis
- 2 Exemples de système réparti
  - Problème de l'élection d'un leader
  - Problème du snapshot
- 3 Systèmes répartis : modèles et langages
- 4 Algorithmes répartis

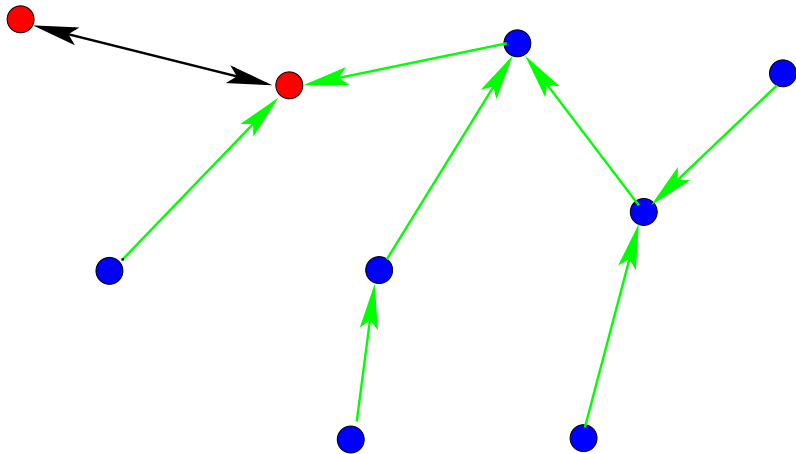
- Les nœuds connaissent uniquement leurs voisins
- La question est d'assurer que chaque nœud sait qu'il est ou non le leader et que ce leader est unique

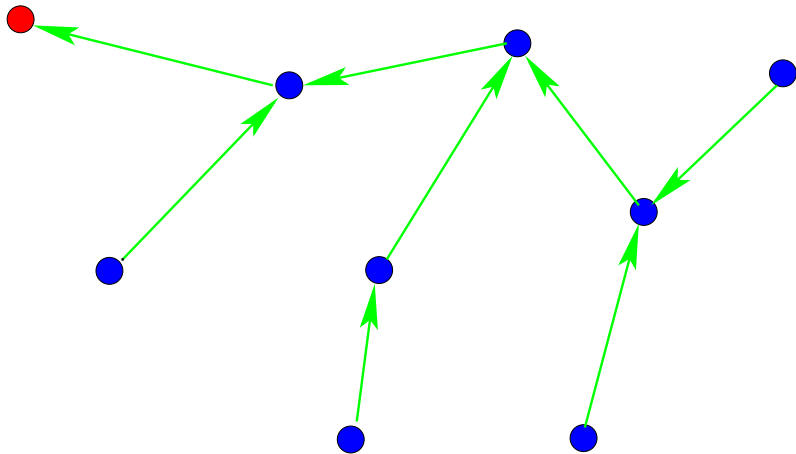






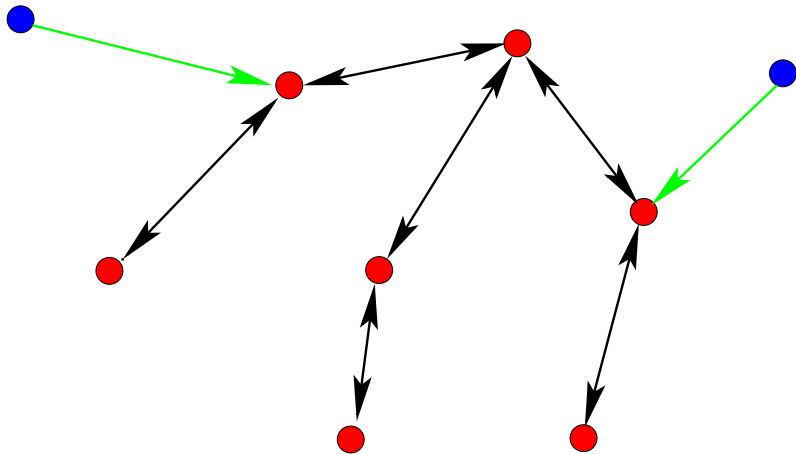




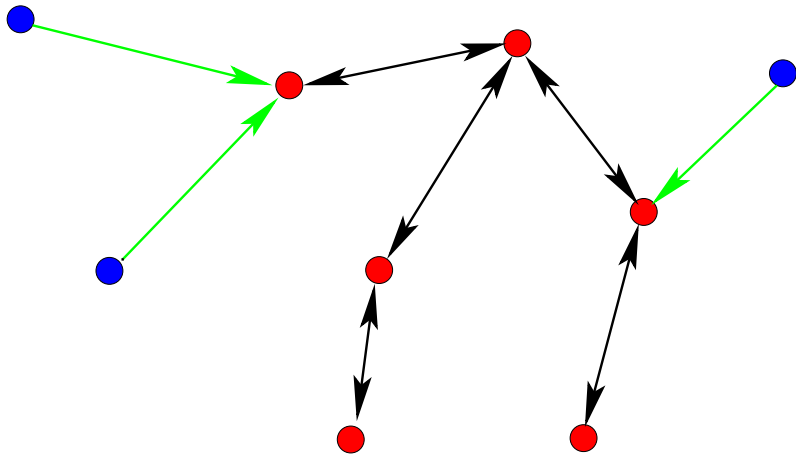




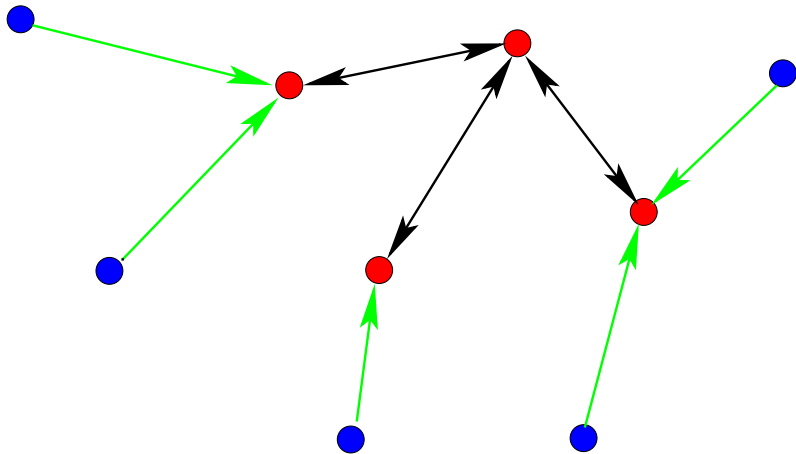


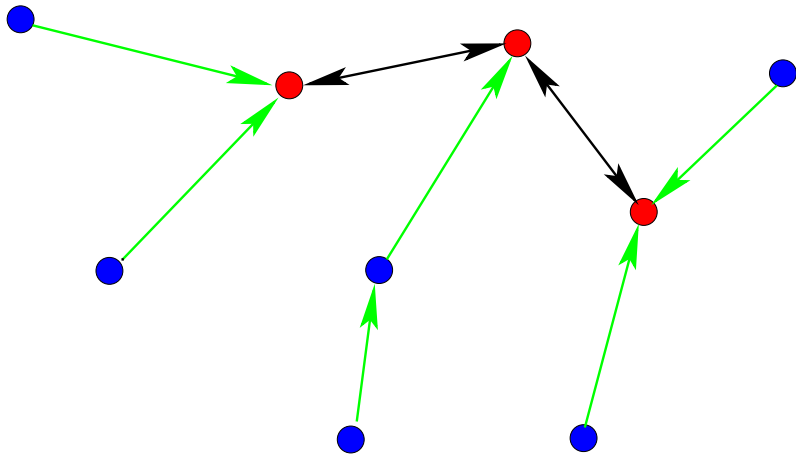


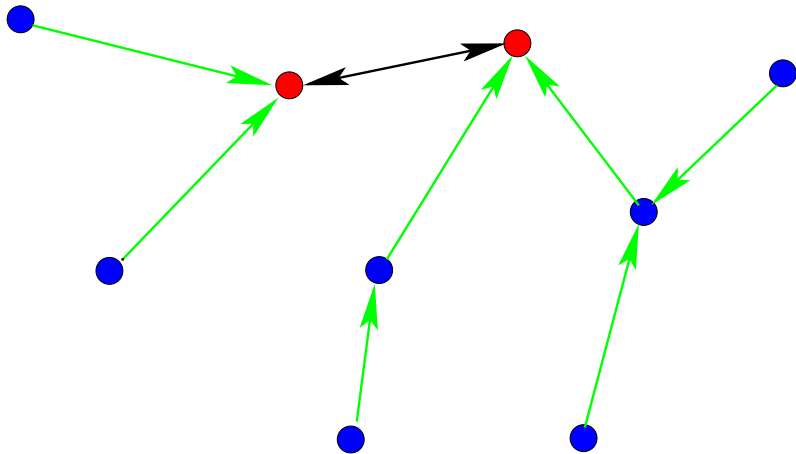






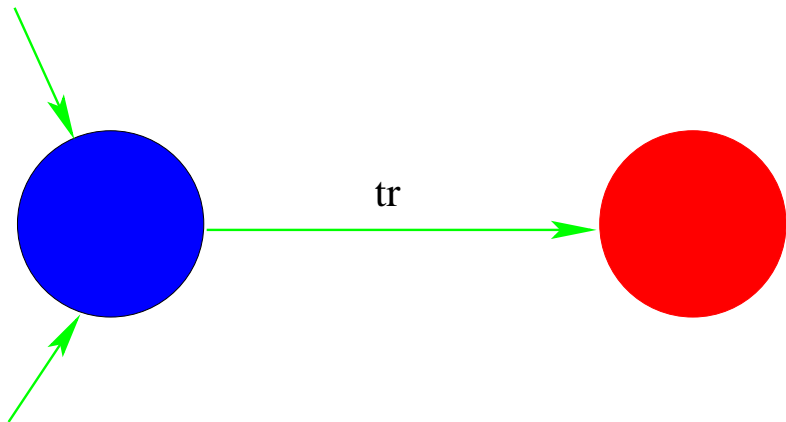












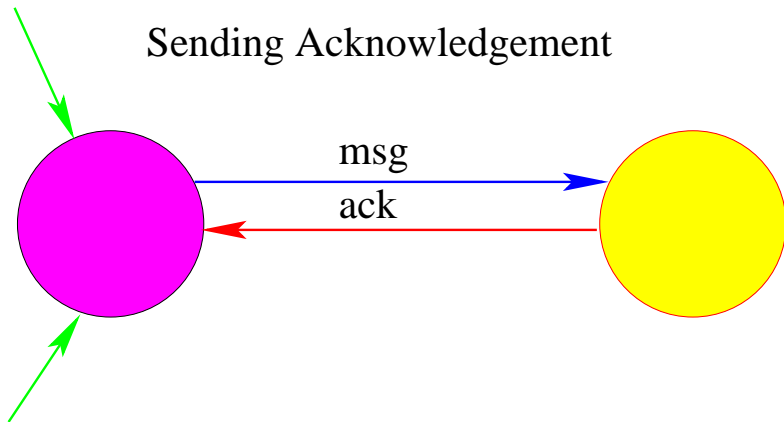




```

graph LR
    In1(( )) -- green --> M(( ))
    In2(( )) -- green --> M
    M -- msg --> R(( ))
    style In1 fill:none,stroke:none
    style In2 fill:none,stroke:none
    style M fill:magenta,stroke:#000,stroke-width:1px
    style R fill:red,stroke:#000,stroke-width:1px
  
```

## Sending Acknowledgement



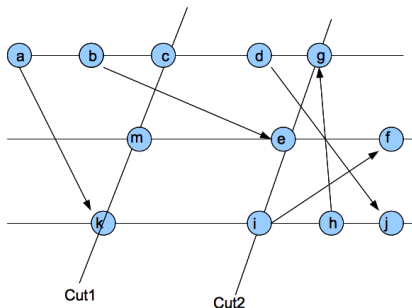
The diagram, titled "Sending Confirmation", illustrates a three-step communication process between a blue circular sender on the left and a yellow circular receiver on the right. Step 1: A blue arrow labeled "msg" points from the sender to the receiver. Step 2: A red arrow labeled "ack" points from the receiver back to the sender. Step 3: A green arrow labeled "tr" points from the sender to the receiver. Additionally, two green arrows point towards the blue sender from the left side of the frame.



- 1 Problèmes des systèmes répartis
- 2 Exemples de système réparti
  - Problème de l'élection d'un leader
  - Problème du snapshot
- 3 Systèmes répartis : modèles et langages
- 4 Algorithmes répartis



# Coups



- $Cut1 = \{a, b, c, m, k\}$
- $Cut2 = \{a, b, c, d, g, m, e, k, i\}$

- Une coupure  $C$  est un ensemble d'événements
- Une coupure consistante  $C$  est une coupure satisfaisant la propriété suivante :  $\forall a, b. a \in C \wedge b < a \Rightarrow b \in C$
- $Cut1$  est consistante mais  $Cut2$  n'est pas consistante ( $h < g$  mais  $h \notin Cut2$ ).



## Des coupures aux snapshots

- Deux coupures peuvent être équivalentes dans la mesure où l'échange de deux actions indépendantes c'est-à-dire causalement indépendantes.
- Pour une coupure donnée, on peut définir l'ensemble des événements les plus récents : ces événements constituent une frontière du calcul en cours.
- Une coupure induit donc un snapshot : une photographie du système et de son état global.
- Toute coupure consistante induit un snapshot consistant et le snapshot courant est l'ensemble des états locaux suivants les événements récemment enregistrés.
- Si  $C1$  et  $C2$  sont deux coupures telles que  $C1 \subseteq C2$ , alors  $S2$  est plus récent que  $S1$





- Sémaphores
- Moniteurs
- Pipes
- Passage de messages

# Systèmes répartis ?

---

- Programmation répartie par envoi de messages synchrone : ADA, CSP, OCCAM
- Programmation répartie avec bibliothèques de communication : PVM, MPI
- Programmation répartie avec coordination : X LINDA
- Programmation répartie mobile : réseaux actifs, codes mobiles
- Programmation répartie objet : Java

## classiques de programmation

- Modèles de calcul classiques :  $\lambda$ -calcul
  - ▶ suites d'opérations
  - ▶ composition de fonctions
  - ▶ entrée-sortie
- Modèles de calcul réparti :  $\pi$ -calcul
  - ▶ suites d'opérations locales et globales
  - ▶ non déterminisme
  - ▶ interaction

- Systèmes événementiels en TLA<sup>+</sup> et PlusCal
- Utilisation d'outils : TLAPS Toolbox

## Problèmes associés aux systèmes répartis

- Protocoles de communication
- Algorithmes d'élection du leader comme IEEE 1394
- Stabilisation des systèmes : auto-stabilisation
- Partage de ressources : exclusion mutuelle
- Détection de la terminaison dans un réseau connexe de machines





# Sommaire du cours

- Modélisation des algorithmes répartis
  - ① Systèmes Répartis : Problèmes - Concepts - Techniques - Outils
  - ② Modélisation des systèmes répartis
  - ③ Calculs Locaux en TLA<sup>+</sup>
  - ④ Utilisation de PlusCal comme notation algorithmique
- Algorithmes répartis selon les services attendus :
  - ① Communiquer avec les protocoles de communications
  - ② Ordonner avec les temps et les datations dans les systèmes répartis.
  - ③ Partager avec les algorithmes d'exclusion mutuelle.
  - ④ Organiser avec les algorithmes d' élection du leader
  - ⑤ Contrôler avec les algorithmes de snapshots et de détection de la terminaison.
  - ⑥ Stabiliser avec les algorithmes d'autostabilisation

# Présentation du cours

- Fondements : modélisation, vérification, analyse, validation, outils.
- Algorithmique : routage, exclusion mutuelle, élection, ondelette.
- Applications réparties
- Utilisation de l'outil TLA Toolbox.
- Organisation : Cours et TD D. Méry

## Contenu du cours

- Développer des modèles réalistes de systèmes.
- Introduire les concepts nécessaires ou les rappeler.
- Expérimenter la démarche dans un outil.
- Gérer au mieux les abstractions et les raffinements.

# Présentation

- Utiliser une technique de modélisation aussi générale que possible
- Un système réparti comprend des nœuds ou des sites répartis physiquement
- Chaque site comprend des actions ou des événements qui ont une action locale ou de communication.
- Un système réparti est modélisé par une relation de transition non-déterministe et avec éventuellement de l'équité
- Les systèmes de transitions sont de bons outils pour modéliser de tels systèmes.

# Sommaire du cours

- Modélisation des algorithmes répartis
  - ① Systèmes Répartis : Problèmes - Concepts - Techniques - Outils
  - ② Modélisation des systèmes répartis
- Algorithmes répartis
  - ① Communiquer avec les protocoles de communications (ABP, STENNING, SLW)
  - ② Ordonner avec les temps et les datations dans les systèmes répartis. (ABCAST,CBCAST)
  - ③ Partager avec les algorithmes d'exclusion mutuelle (BAKERY, RICARTAGRAWALA, CARVALOROUCAIROL MAEKAWA)
  - ④ Organiser avec les algorithmes d' élection du leader ( LELANN, IEEE1394)
  - ⑤ Contrôler avec les algorithmes de snapshots, de détection de la terminaison et les arbres de recouvrement.
  - ⑥ Stabiliser avec les algorithmes d'autostabilisation.