Vérification des Systèmes Mobiles (VSM)

Master 2 / GLSD

2016-2017

Plan du Cours

- **Définitions de base**
- **Motivations de la mobilité**
- **Modèle d'un système de code mobile**
- **Mécanismes pour la mobilité/ Stratégies pour la mobilité**
- **Technologies pour la mobilité (langages de programmation, Environnement de développement)**
- **Exemples de réalisations et de grands projets industriels**
- **Génie logiciel pour les systèmes mobiles (Approches de Développement, Langages pour la spécification)**

Mobilité en Informatique concepts

- *mobile computing*,
- *mobile code (ou code mobility)*
- et *mobile agent*.

Mobilité en Informatique

mobile computing vs code mobility

Mobile computing :
"*Paradigm in which users carrying portable devices have access to a shared infrastructure independent of their physical location*".**[Geo+96]**

Code Mobility:
"Code mobility can be defined as the ***capability to dynamically change*** *the* ***bindings*** *between* ***code fragments*** *and the* ***location*** *where they are executed"* **[Car+97]**.

*"****Mobile code*** *are soft-ware that* ***travels on a heterogeneous network****, crossing administrative domains, and is* ***automatically executed upon arrival*** *at the destination …"***[Tho 97]**

Mobilité en Informatique

mobile Agents

*"Mobile agents **are programs** that **can move** through a network under their **own control**, **migrating** from host to host and **interacting** with other agents and resources on each"* **[Gray+96]**

*"A mobile agent is **not bound** to the system where it begins execution. It has **the unique ability to transport** itself from one system in a network to another.* **[Lan 98]**

*"Computations that **are able to relocate themselves** from one host to another "* **[Nil+03]**

Mobilité en Informatique

Mobile Agents

*"A **piece of code** and its **associated data** moving about **executing autonomously** on behalf of its owner"* [**Joh 04]**

*"Mobile agents are **software abstractions** that can **migrate** across the network **representing users** in various tasks… A mobile agent has the **unique ability to** transport itself from one system in a network to another in the same network"* **[Bar 05]**.

Mobilité en Informatique en résumé

Deux classes

- **Mobilité physique**
- **Mobilité logicielle**

Pourquoi la mobilité ?

Mobilité Physique

1) La diffusion des réseaux informatique sans fils,

2) La diffusion des réseaux de télécommunication cellulaires.

3) Les utilisateurs mobiles

Pourquoi?
Mobilité logicielle

- Un **équilibrage de charge** entre processeurs.
- Une **performance dans la communication**. En rassemblant les objets qui se communiquent intensivement sur les même nœuds.
- **Disponibilité des objets** requis chez les nœuds demandant ces objets.
- Une obtention (par téléchargement) de **services logiciels disponibles** sur un certain nœud.

Pourquoi?
Mobilité Logicielle

- **Gestion flexible de Données** : Les nœuds d'un réseau reçoivent les données ainsi que les protocoles de leurs traitements (encapsulation de protocoles).

- Améliorer la **tolérance aux fautes** dans les applications distribuées

Pourquoi?
Mobilité logicielle

- La nature **dynamique des infrastructures** de télécommunication : débit faible et une faible fiabilité (déconnexions fréquentes).

- Réalisation **d'applications autonomes** : de telles applications se déplacent sur le net, pour assurer des **interactions locales complexes.**

Pourquoi?
Mobilité Logicielle

- **Déploiement et maintenance** de composants de systèmes distribués sur les réseaux à grande échelle : Des composants pour la maintenabilité et la reconfiguration, se déplacent sur le réseau et visitent les différents nœuds pour faire le nécessaire.

- **Besoins de spécialisation de services** : Le nombre de clients croît rapidement ainsi que leurs besoins se différencient. Les serveurs prédéfinis avec un minimum de services peuvent être spécialisés via leur enrichissement par des composants (en les téléchargeant) assurant des services spécifiques.

Pourquoi?
Mobilité Logicielle: Agents Mobiles

- Ils **s'exécutent de façon autonome et asynchrone** : pour les dispositifs mobiles avec des **fréquentes déconnexions**,
- les agents mobiles peuvent **migrer en d'hors de ces dispositifs lors de la connexion**, durant une déconnexion,
- ces agents **s'exécutent sur le réseau de manière asynchrone et autonome**.
- En terminant leurs missions, **ils reviennent aux dispositifs initiaux** lors des prochaines connexions.

Pourquoi?
Mobilité Logicielle: Agents Mobiles

Réseau

Dispositif avec une faible connexion

Agent logiciel

Communication intensive+
faible débit+
déconnexion fréquente

Réseau

Agent logiciel

Réseau

Agent logiciel

Pourquoi?
Mobilité Logicielle (Agent)

- Ils **s'adaptent dynamiquement** : ils **sentent** les propriétés de leur environnement (les segments du réseau non fiables, les segment où le débit est faible, …) pour se reconfigurer (se dispatcher) afin d'avoir une configuration optimale (équilibrer la charge, …)

Pourquoi?
Mobilité Logicielle (Agents)

- Ils **sont hétérogènes** de nature : ils sont **indépendants des hôtes** et de moyens de transport. Ils dépendent **seulement de leur environnement logiciel d'exécution** (exp: JVM).

Pourquoi?
Mobilité Logicielle (Agents)

- Un paradigme plus facile à comprendre et à utiliser pour concevoir des systèmes compliqués

Remarque

Le reste du cours s'intéresse à la mobilité logicielle et aux agents mobiles

Origines de l'Idée

- Le langage **d'impression PostScript** **[Ado 85]:** Pour imprimer sur une imprimante PostScript, on exige un programme PostScript, décrivant toutes les propriétés de la page à imprimer, sera envoyé à l'imprimante, où il sera exécuté.

- le langage ***distributed smalltalk*** **[Dou 87]**,

- La **migration transparente de processus et d'objets actifs** dans les systèmes distribués

  Accent **[Ras+81]**, *Eden* **[Laz+81]**, *DEMOS/MP* **[Pow+83]**, Emerlad **[Jul + 88]**, Chorus **[Roz + 88]**, Locus **[But+84**, **Thi 91]**, *V-ystem* **[The+85]**, et Cool **[Lea + 93]**)

Quelle Nouveauté Donc?

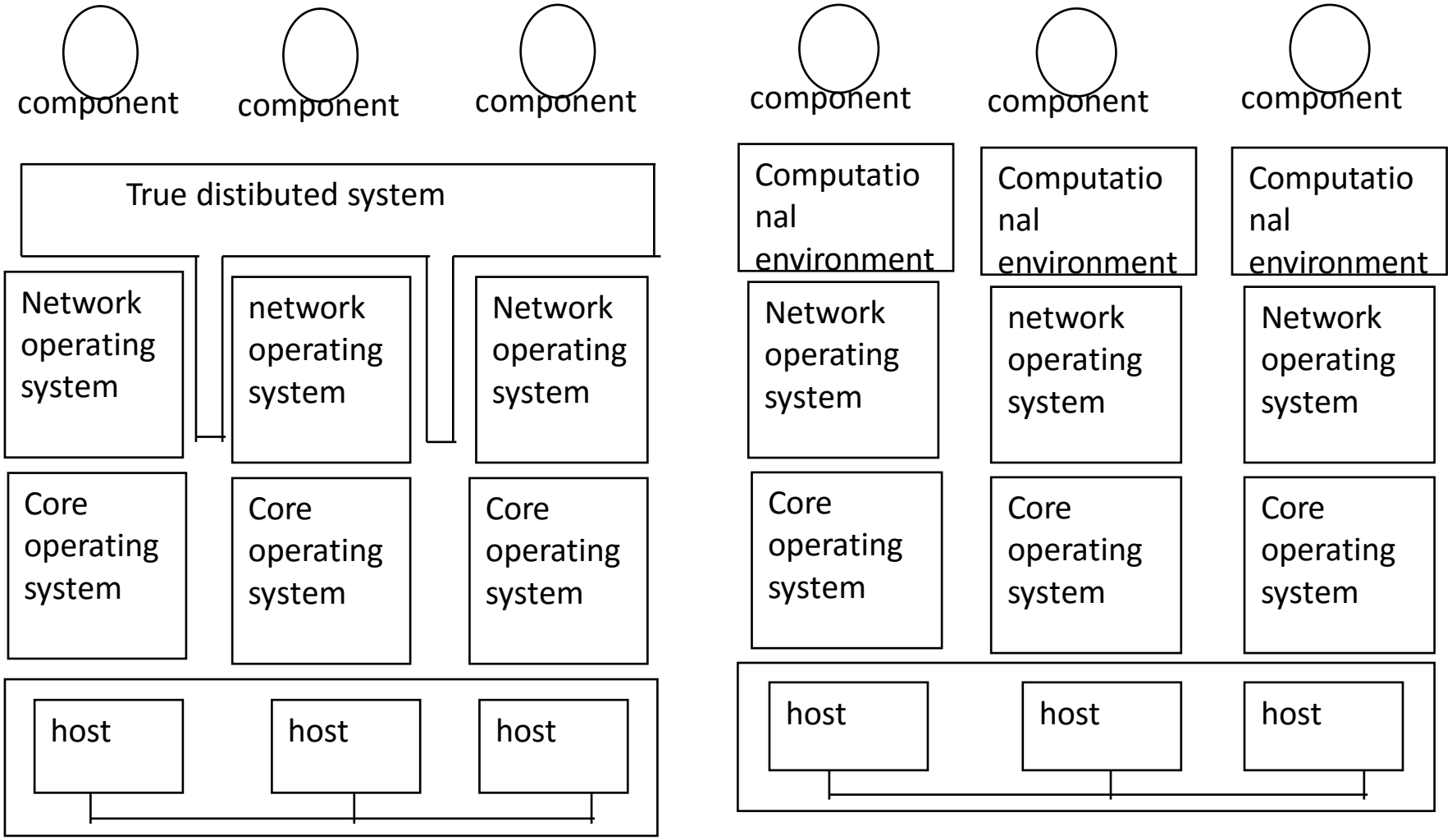
- On s'intéresse au **système à grande échelle de type Internet (WAN),** et non plus aux LANs des systèmes distribués classiques.

- La mobilité est **sous le contrôle du programmeur** (*Programming is location aware*). Elle est non transparente.

- L'objectif de la mobilité **n'est plus uniquement l'équilibrage de charge**.

Système de Code Mobile (MCS)

Architectures pour MCS
modèle pour MCS



Architecture pour MCS
modèle pour MCS

- La couche 1 : « *hardware* »: hôtes (nœuds) + l'infrastructure du réseau de communication.

- Couche 2 : « *core operating system* », le noyau du système d'exploitation de pour chacune des hôtes. Cette couche doit assurer les fonctions de bases : gestion de fichiers, de mémoire et de processus.

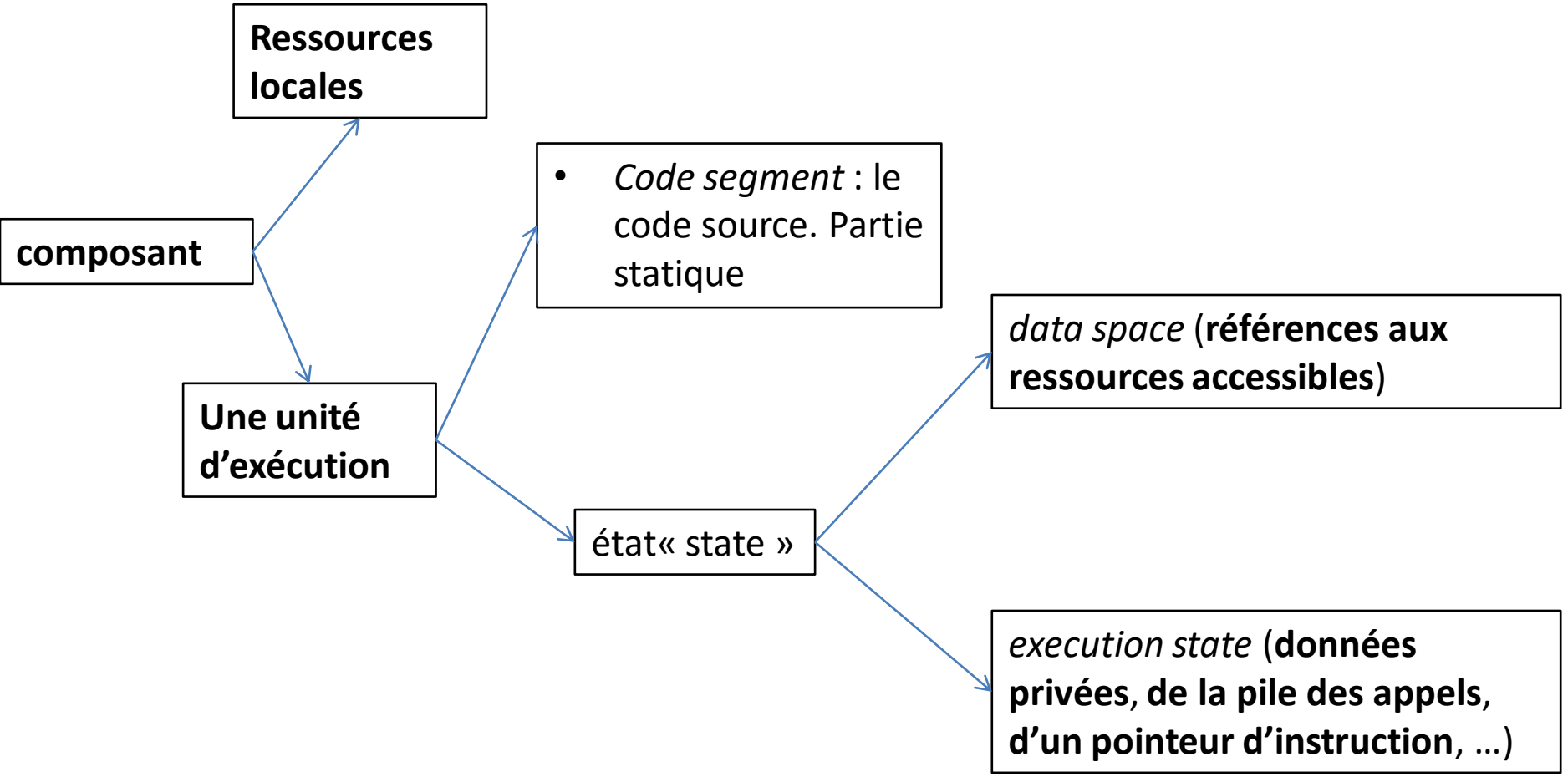
- Couche 3 : « *network operating system* »: Dans cette couche, les communications sont non transparentes.

Architecture pour MCS
modèle pour MCS

- Couche 4 : fournir le service de mobilité. Elle peut être soit transparente « *true distribuetd system* », ou bien un « *computaional environment* » où les communications et la mobilité sont non transparentes.

- Couche 5 : « ***component*** », englobe soit **des ressources** (logique ou physique) **et des calculs** (programmes, processus ou applications) nommés « ***executing units* : EUs** ».

Architecture pour MCS
modèle pour MCS



Architecture pour MCS
modèle pour MCS

- Selon cette structure de l’ EU, on distingue **une variété de techniques** pour la mobilité de code et aussi **une variété de méthode pour gérer les liens** entre le code et les ressources utilisées par ce code après la migration

Architecture pour MCS
Types de mobilité

1) **Mobilité Forte : Mobilité du code + état**

2) **Mobilité Faible : Mobilité du code seulement**

Dans ces deux grande classe, on peut rencontrer différente technique aussi

Architecture pour MCS
Mobilité Forte

Deux types de mobilité forte

1) **La migration**: l'EU est suspendu sur le premier CE, envoyé vers le 2 CE, ou elle continue son exécution. Deux possibilité :

   -**proactive**: la destination et le moment sont définis par l'EU migrante

   -**réactive**: la destination et le moment sont définis par une autre EU

2) **Le clonage à distance**: une copie de l'EU est créée sur un autre CE. Le clonage peut être proactif ou réactif.

Architecture pour MCS
Mobilité Faible

La mobilité faible est aussi classifiée en plusieurs classes :

1. ***Shipping*** : Le CE source envoie le code vers un CE destination. Dans cette classe, on peut trouver (idem pour 2):
   1.1. ***stand-alone code*** (exécution à la volée) : Le code transféré crée sa propre EU pour son exécution sur le CE destination. Dans ce cas : le CE source est le CE destination peuvent être (idem pour 1.2) :

      1.1.1. **synchrones** : le CE source est suspendu en attendant la terminaison de l'exécution du code transféré dans le CE destination.

      1.1.2. **asynchrones** : le CE source n'est pas suspendu. Dans ce cas, le code transféré peut commencer son exécution sur le CE destination :

         1.1.2.1. immédiatement.

         1.1.2.1. différemment : en attendant un événement ou une condition, pour se lancer.

   1.2**.** ***code fragment*** : l'exécution du code transféré se réalise dans le contexte d'un EU existant sur le CE destination.

Architecture pour MCS
Mobilité Faible

- 2. ***Fetching*** : un CE (destination) télécharge un code chez lui.

Architecture pour MCS
types de références avec les ressources

- **Par identité** : c'est le lien le plus fort. L'EU doit avoir toujours accès à cette ressource pour qu'elle s'exécute.

- **Par valeur** : l'EU porte intérêt au contenu de la ressource et non à la ressource elle-même. S'il est possible d'avoir une copie du contenu de cette ressource, l'EU sera satisfait.

- **Par type** : ce qui est important est le type de la ressource. Une ressource de même type peut satisfaire l'EU.

Mécanismes de gestion de liens avec les ressources

Type de ressource / Type de lien	transférable	Non transférable
Par identité	*By move* et *Network reference*	*Network reference*
Par valeur	*By copy (*ou *by move* ou *network reference)*	*Network reference*
Par type	*Rebinding* (ou l'un des autres mécanismes)	*Rebinding* (ou *network reference*)

Mécanismes pour la mobilité de code