LES INTERGICIELS DE RÉPARTITION

1. **Rôle de l'intergiciel**

La mise en œuvre d'un système réparti suppose l'existence de logiciels qui fournissent aux composants applicatifs les abstractions d'un ou plusieurs modèles de répartition.

Un intergiciel fournit au développeur d'application répartie, les moyens de faire interagir des composants distants sans se préoccuper des tâches liées à la répartition.

Il s'agit donc d'une solution logicielle chargée de mettre en place une couche d'abstraction de haut niveau dans le but de simplifier le développement et le fonctionnement d'applications distribuées.

Un intergiciel simplifie le développement d'applications réparties en prenant en charge certains aspects de la répartition : gestion des communications entre les noeuds, hétérogénéité des architectures matérielles et logicielles, apport de nombreux services de haut niveau.

Tout doit se passer comme si l'application se déroulait en local...

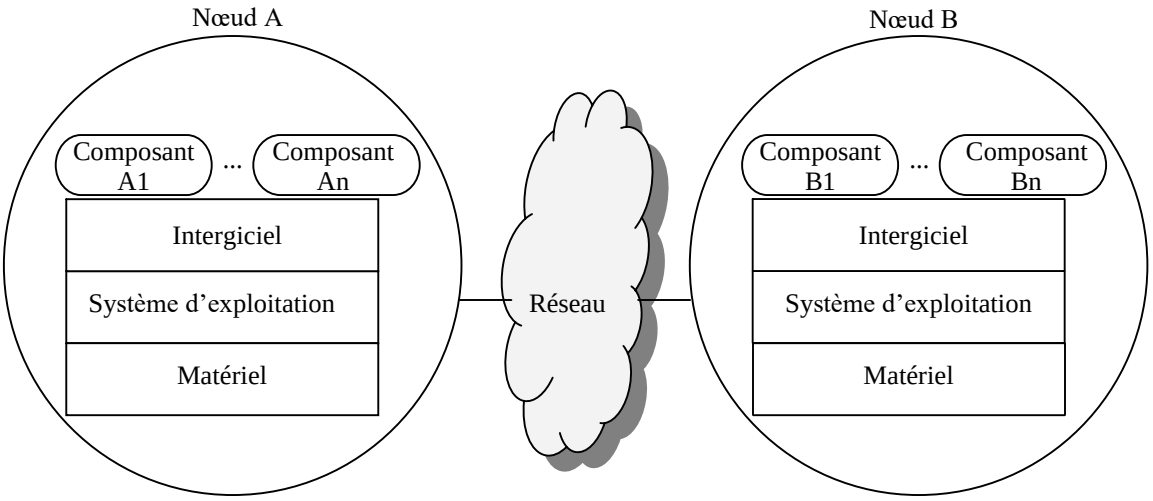

Fig.1 : structure d'une application répartie utilisant un intergiciel

1.1 Transparence vis à vis de la localisation des composants :

Pour le développeur d'une application utilisant un modèle de répartition de haut niveau (appel de sous-programmes à distance, objets répartis ou mémoire partagée, par exemple), la répartition est introduite de façon « transparente », dans la mesure où tout se passe comme si l'application se déroulait localement.

La répartition consiste alors à projeter les entités ainsi délimitées sur les différents noeuds participant à l'application répartie.

Les échanges de messages sous-tendus par le modèle de répartition sont effectués par l'intergiciel.

1.2 Independance vis à vis de la plate-forme d'exécution :

Les composants de l'application utilisent les fonctions de l'intergiciel pour communiquer entre eux de façon transparente ; l'intergiciel réduit la dépendance des composants vis-à-vis de l'environnement d'exécution, en masquant les aspects d'interfaçage de bas niveau avec des primitives de communication de haut niveau.

Chaque nœud possède donc une instance de l'intergiciel. Ces instances communiquent entre elles au moyen de primitives de bas niveau, et jouent le rôle d'intermédiaires entre les composants de l'application qui résident sur le même nœud et ceux qui résident sur d'autres nœuds. Les interactions avec ces derniers sont réalisées par échange de messages entre les instances d'intergiciel locale et distante.

2. **Fonctions de l'intergiciel**

... ce qui suppose que le Middleware fournit des fonctionnalités qui « masquent » la répartition…

2.1 Adressage/nommage

L'intergiciel doit avant tout définir un moyen de nommer les diverses entités qui participent à une application répartie : il leur attribue des identifiants susceptibles d'être échangés et compris par chacun des nœuds de l'application. L'intergiciel remplit une fonction d'adressage des composants de l'application.

2.2 Transport

L'intergiciel assure le transport de messages entre les nœuds, en utilisant le réseau de communication sous-jacent.

2.3 Protocole et représentation

L'intergiciel met en œuvre un ou plusieurs protocoles de communication et se charge également de mettre les données à échanger sous une forme transmissible, et compréhensible par les nœuds distants, c'est-à-dire qu'il applique une syntaxe et une sémantique aux messages.

Protocole et représentation sont indépendants d'un langage de programmation ou d'une architecture matérielle. L'utilisation d'un intergiciel permet donc de faire interagir, au sein d'une même application répartie, des composants s'exécutant sur des architectures matérielles et dans des environnements logiciels différents.

En contrepartie, le choix d'un protocole et d'une représentation limite les interactions possibles aux composants dont l'intergiciel supporte ce protocole.

2.4 Liaison :

Un objet qui dispose d'une référence désignant un autre objet, à distance, ne peut directement en appeler les méthodes. Il est nécessaire, au préalable, que l'intergiciel fasse la liaison entre la référence d'objet et les ressources nécessaires à l'appel de méthodes sur l'objet distant.

2.5 Activation/ cycle de vie :

L'intergiciel doit s'assurer que la structure concrète associée à un identifiant objet existe bien et que l'association entre les deux composants soit maintenue. On désigne cette fonction de contrôle du cycle de vie des associations entre identifiants et composants applications par le terme d'activation.

2.6 Exécution :

L'intergiciel permet la synchronisation des échanges et des traitements. Là encore, il reflète le modèle de répartition mis en œuvre :

- Exécution synchrone : un composant qui demande à un autre d'exécuter un service, attend la fin de l'exécution de ce service (matérialisée par la réception d'un message) avant de poursuivre son déroulement.
- Exécution asynchrone : un composant qui demande à un autre d'exécuter un service, poursuit son propre traitement pendant que le second composant traite sa demande.

3. Limites introduites par l'intergiciel

L'utilisation d'un intergiciel dans une application répartie, par opposition à l'utilisation directe des primitives de communication fournies par le système d'exploitation, permet l'utilisation d'un modèle de répartition sophistiqué. Ce gain en abstraction facilite également la mise en place d'applications hétérogènes.

Toutefois, les apports des intergiciels s'accompagnent de plusieurs contreparties.

3.1 Coûts relatifs des différents modèles de répartition

À chaque modèle de répartition correspond un coût de mise en œuvre.

Plus les fonctionnalités offertes sont importantes, plus l'intergiciel est volumineux, et plus les ressources nécessaires sont considérables.

Un intergiciel offrant le modèle « passage de messages » peut n'introduire qu'un surcoût minime par rapport à l'utilisation directe des primitives de communication du système d'exploitation sous-jacent.

En effet, il rend un service d'un faible niveau d'abstraction, dont la réalisation transparente ne nécessite pas d'opérations complexes.

Lorsqu'un modèle plus évolué (appel de sous-programmes à distance ou objets répartis) est mis en œuvre, en revanche, l'intergiciel effectue des traitements complexes. Il occupe donc plus d'espace en mémoire. Les fonctions avancées sont également susceptibles d'introduire des délais de traitement supplémentaires pour tous les messages échangés, et d'augmenter le volume de données à transmettre entre les nœuds.

3.2 choix d'un modèle de répartition :

Les intergiciels sont liés au choix d'un modèle de répartition en général, et au choix de leur mise en oeuvre en particulier. Deux mises en œuvre du même modèle ne sont pas nécessairement compatibles.

Cette contrainte tendant à imposer le choix d'un seul intergiciel, pour l'ensemble d'une application, s'oppose à l'interopérabilité entre composants divers, et à fait naître le besoin d'interconnecter les intergiciels par des passerelles réalisant l'adaptation entre modèles de répartition. Celles-ci fondent la notion de communication entre intergiciels, ou « M2M » (Middleware to Middleware).

4. Exemples :

- CORBA et son sous-système de communication ORB.
- COM+ (Common Object Model) et son sous-système de communication DCOM (Distributed Common Object Model) dans le monde Microsoft.