UNIVERSITE BADJI MOKHTAR-ANNABA
FACULTE DES SCIENCES DE L'INGENIORAT
DEPARTEMENT D'INFORMATIQUE

جامعة باجي مختار – عنابـــــــــــة
كلية علـــــــــوم الهندســـــــة
قسم الإعـــــــلام الألـــــــــي

SYSTEMES DISTRIBUES ET SERVICES WEB

Programme de Première Année Master en Informatique
Option Ingeniérie des Logiciels Complexes

SUPPORT DE COURS REALISE PAR
PR DJAMEL MESLATI

VERSION 2016

CHAPITRE 1

INTRODUCTION AUX SYSTEMES DISTRIBUES

CONTENU

1.1 Définitions et caractéristiques
Concurrence, partage de ressources, communication par message, absence d'horloge commune, indépendance des pannes

1.2 Exemples de systèmes distribués
Internet, intranet et systèmes mobiles

1.3 Problèmes
L'hétérogénéité, la concurrence, la sécurité, les pannes, absence d'informations globales sur un système distribué

1.4 Défis et objectifs
L'interopérabilité, l'ouverture, l'invariance à l'échelle (scalability), l'efficacité, la disponibilité, la gestion de la sécurité, le maintien de la consistance des ressource, la transparence (flexibilité), la gestion des situations d'exception (détection des pannes et reprise), la gestion des transactions réparties.

1.1 DEFINITIONS ET CARACTERISTIQUES

A. Tanenbaum et M. Van Steen [Tan 2002], donne la définition suivante : Un système distribués est une collection d'ordinateurs indépendants qui apparaissent à l'utilisateur comme un seul système cohérent.

De cette définition ressortent deux aspects : Le premier est de nature matérielle, il s'agit de l'autonomie des ordinateurs ou chacun peut exécuter des tâches en concurrence (c à d au même moment) avec les autres. Le second fait référence au logiciel qui laisse apparaître le système distribué comme une seule entité cohérente. Bien entendu, le second aspect suppose une interconnexion des ordinateurs moyennant un réseau de communication physique.

Selon G. Coulouris et al [Cou 2001], Un système distribué ou réparti est un système dont les composants sont répartis sur différents nœuds d'un réseau d'ordinateurs. Ces composants communiquent et coordonnent leurs actions uniquement par l'échange de messages.

Les messages étant des suites de bits représentant des informations qui ont un sens pour l'émetteur et le récepteur. D'une manière plus formelle, les systèmes répartis se caractérisent par :

- Une absence de base de temps commune (horloge commune)
- Une absence de mémoire commune semblable à celle d'un système multiprocesseur
- Les pannes des composants sont indépendantes

La motivation principale des systèmes répartis est l'amélioration du partage des ressources. Autrement dit, les ressources disponibles dans un ordinateur donné peuvent être utilisées par un autre et vice versa. Par exemple, une imprimante connectée à un ordinateur peut recevoir des informations à imprimer provenant d'un autre ordinateur. De même, les fichiers disponibles dans un répertoire d'un ordinateur donné peuvent être manipulés par les autres ordinateurs, etc.

Naturellement, ce partage de ressources entre tâches concurrentes introduit des problèmes souvent désignés par « Problèmes de concurrence ». Ces problèmes existent aussi dans les systèmes centralisés multitâches (systèmes multiprocesseurs ou monoprocesseur avec partage de temps). Cependant, dans un système distribué, les problèmes de concurrence se trouve accentués par le fait que la gestion des ressources n'est pas centralisée et la nécessité d'opérer des communications entre ordinateurs, ce qui exige des techniques de gestion plus sophistiquées.

La communication dans les systèmes distribués s'opère entre entités physiques ou logiques et se base sur la possibilité d'échange de messages qui est toujours supportée par le réseau physique d'ordinateurs. Par exemple, l'échange entre deux cartes de communication réseau est un échange entre entités physiques. L'échange entre deux systèmes de gestions de fichiers est un échange entre entités logiques. Il en est de même pour la communication entre applications où l'échange s'opère entre entités logiques appelées processus. Il existe diverses approches de communications allant d'un simple échange de bits à des protocoles complexes tel que l'invocation de méthodes à distance (couramment nommé RMI qui est une abréviation de Remote Method Invocation) ou la communication par événements. Notons cependant que l'échange entre entités logiques passe par l'échange entre entités physiques.

L'absence d'horloge commune citée précédemment comme caractéristique des systèmes répartis, signifie que seule la communication de messages peut être utilisée pour coordonner les activités des différentes tâches du systèmes et garantir la cohérence lors du partage des ressources. En effet, il n'est pas possible de réaliser une synchronisation des horloges des différents ordinateurs et coordonner par la suite les tâches en fonction d'un référentiel temporel unique.

Dans un système distribué, les ordinateurs peuvent tomber en pannes indépendamment les uns des autres. De même, les lignes de communication physiques peuvent être défaillantes. Mais au-delà des aspects matériels, les tâches exécutées sur chaque ordinateur peuvent échouées. Ces pannes d'ordre physique ou logique peuvent

avoir un effet local plus ou moins étendu. Par exemple, une panne au niveau d'un réseau de communication engendre l'isolation des ordinateurs qui y sont connectés sans pour autant arrêter l'exécution sur ces derniers. Les programmes (processus) sur ces ordinateurs peuvent ne pas être capables de détecter si le réseau est en panne ou s'il est simplement devenu lent. De même, l'échec d'une activité quelconque sur un ordinateur n'est pas directement détectable par les autres ordinateurs sur le réseau. Cependant, une des caractéristiques des systèmes distribués est l'indépendance de ces pannes.

Lorsque, une application fait intervenir plusieurs composants où chacun s'exécute sur un ordinateur séparé, il incombe au concepteur de cette application d'anticiper les différentes pannes qui peuvent arrivées et prévoir des mécanismes pour garantir la poursuite de la coordination et l'atteinte de l'objectif visé.

1.2 EXEMPLES DE SYSTEMES DISTRIBUES

Les exemples des systèmes distribués sont nombreux et variés. Un même système distribué peut être fondé sur plusieurs autres systèmes distribués de niveaux différents. En effet, une application distribuée est un système composé de processus qui s'exécute sur plusieurs ordinateurs connectés par un réseau. Cet ensemble d'ordinateurs connectés est lui-même un système distribué. Les couches des systèmes d'exploitation qui assurent le maintien du réseau constituent un système distribué, etc. Comme exemples illustratifs, nous considérons Internet, les intranets et les systèmes mobiles.

1.2.1 Internet

Internet est une vaste collection de réseaux d'ordinateurs interconnectés. Les programmes qui s'exécutent sur ces ordinateurs interagissent par l'échange de messages en utilisant un moyen de communication ou un autre. La mise en œuvre des moyens de communication (mécanismes et protocoles) constitue un défi considérable qui a permit a un ordinateur quelconque d'échanger des messages avec n'importe quel autre ordinateur dans le réseau.

Internet constitue actuellement le système distribué le plus large au monde et ceci sur quasiment tous les plans. Des utilisateurs de n'importe quelle position dans le monde peuvent utiliser les services offerts par le WWW (World Wilde Web), le FTP (File Transfert Protocole), et autres applications. Les services disponibles peuvent être étendus librement et le système peut être agrandi par l'ajout d'ordinateurs à n'importe quel moment.

Internet a une structure composée d'intranets connectés par des backbones. Ces derniers étant des moyens de communication de haute capacité (utilisant des satellites, la fibre optique, etc.). Les intranets sont la propriété de compagnie et d'organisations

diverses. Les fournisseurs des services Internet (Internet Service Providers) sont en fait des compagnies qui offrent des connexions aux utilisateurs d'Internet (personne ou organisation) par des modems et d'autre dispositifs qui leurs permettent l'accès aux services d'internent aussi bien que l'accès à des services locaux tels que l'hébergement de pages WEB.

Bien que couramment confondus, Internet se distingue du WWW. Le World Wilde Web n'est autre qu'une application parmi tant d'autres que Internet supporte. Par exemple, le FTP est une autre application très utilisée. De même, l'échange Peer-to-Peer (poste à poste) connaît actuellement un succès grandissant. Le WWW représente un système distribué logique consistant en un nombre considérable de ressources (pages Web, fichiers de données et services) référencées par des URL (Uniform Resource Locator). Les pages Web contiennent des liens (des URLs) qui référencent des ressources et d'autres pages Web. L'ensemble des pages Web constitue une toile immense interconnectée par des liens qui permettent de naviguer d'une page à l'autre et de déclencher l'exécution de services divers. Le WWW est supporté par des standards et normes tels que http et Html qui régissent sa structure et le comportement des logiciels qui le supportent tels que les serveurs http et les navigateurs internet.

La figure 1.1 donne un aperçu sur l'architecture Internet

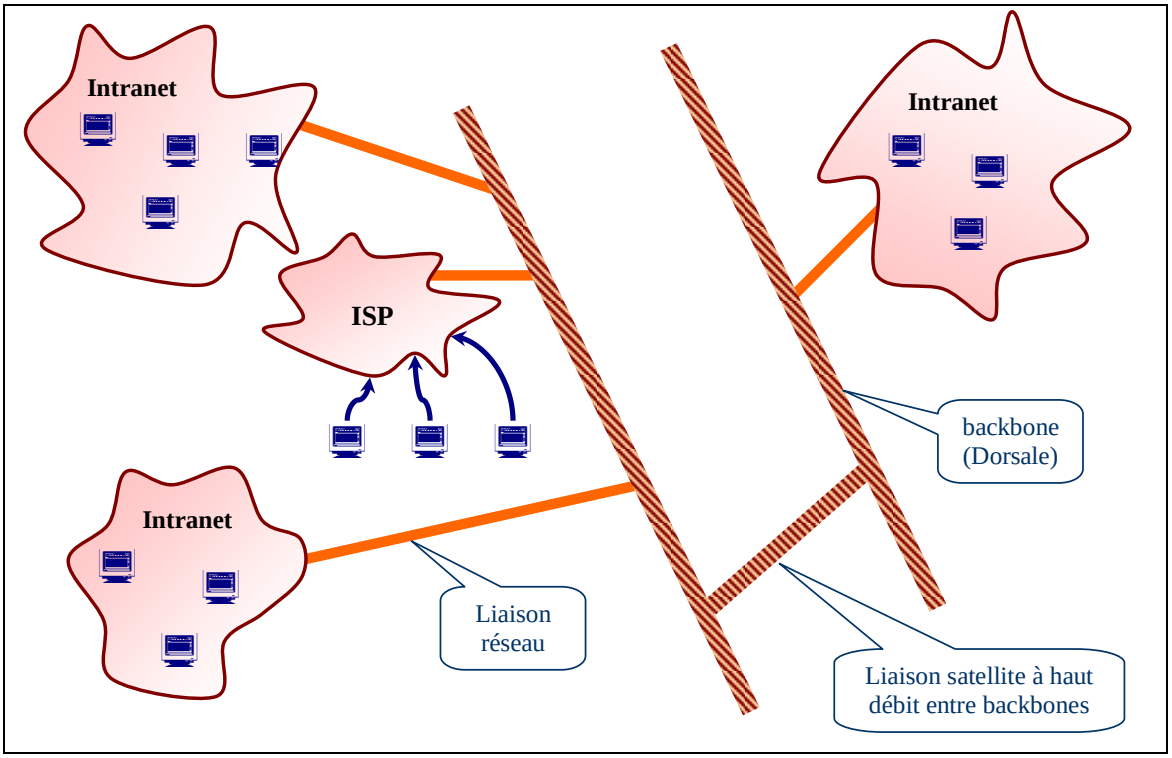

Figure 1.1. Structure d'un fragment Internet

1.2.2 Intranet

L'intranet est une portion Internet administrée séparément et délimité de telle sorte à garantir une politique de sécurité locale.

L'intranet est typiquement composé de plusieurs réseaux locaux (LAN ou Local Area Networks) reliés par des backbones. Chaque réseau local dispose de ressources gérées par des serveurs. La configuration d'un intranet est à la charge de l'organisation qui l'administre et peut varier largement.

Les intranets se connectent à Internet via des routeurs qui permettent aux utilisateurs d'avoir accès aux services Internet mais aussi d'avoir accès aux services d'autres intranets.

Les organisations qui administrent les intranets se protègent des menaces extérieures en utilisant des Firewalls. Ces derniers sont des programmes qui filtrent les données en entrée et en sortie en fonction des politiques de sécurité adoptées par les organisations.

La figure 1.2 donne un exemple de structure d'intranet.

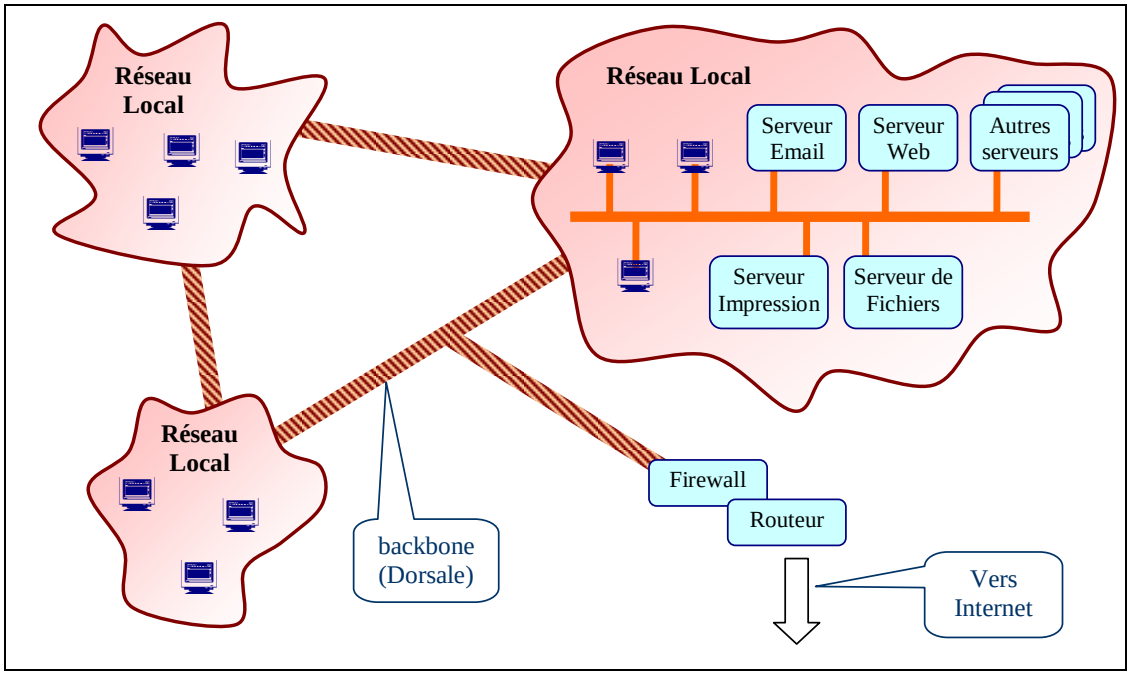

Figure 1.2. Structure typique d'un intranet

1.2.3 Les systèmes mobiles

Le progrès dans la miniaturisation et les réseaux sans fil laisse envisager le concept de *l'espace d'information totalement connecté* (fully connected information space) où chaque entité dotée d'une certaine fonctionnalité est connectée aux autres entités moyennant des dispositifs électroniques et des liaisons sans fil (voir figure 1.3). Ces liaisons permettent à chacune d'avoir des informations sur les autres et éventuellement invoquer les services qu'elles offrent. Comme la majorité des entités sont dynamiques ont a affaire à des systèmes mobiles. Le partage de ressources dans de tels systèmes est une tâche complexe qui nécessite une certaine forme d'organisation et des mécanismes et protocoles ad hoc.

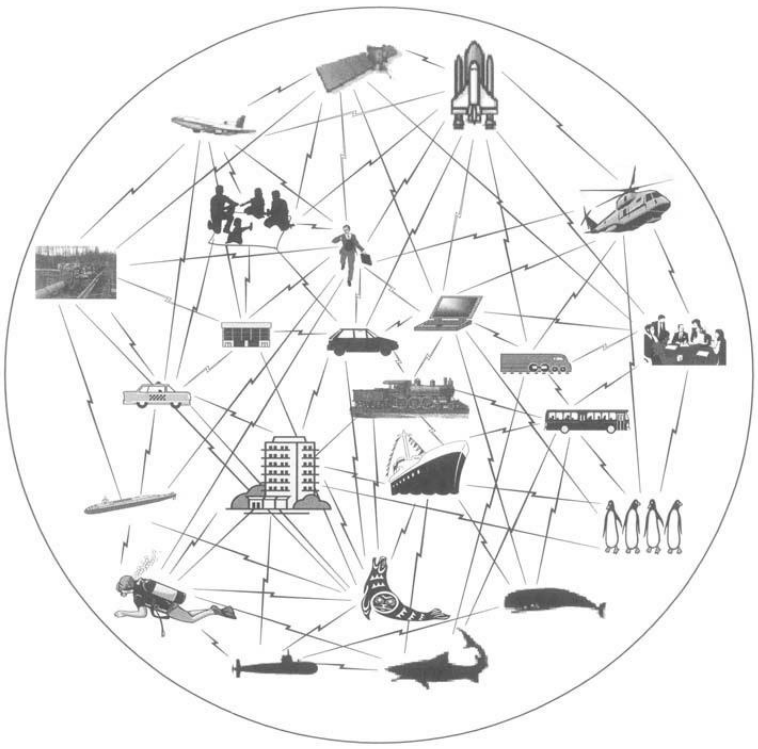

Figure 1.3. Espace d'information totalement connecté

La figure 1.4 donne un exemple d'organisation où un utilisateur visitant une organisation quelconque, accède à son intranet (qualifié de Host) via un réseau sans fil et utilise les ressources et services disponibles (utilisation de l'imprimante locale, accès à des bases de données, …). Ce même utilisateur, a aussi la possibilité d'utiliser, à distance, l'intranet de son domicile via un téléphone portable. Cet exemple illustre une partie de la diversité des possibilités qu'offrent les systèmes mobiles.

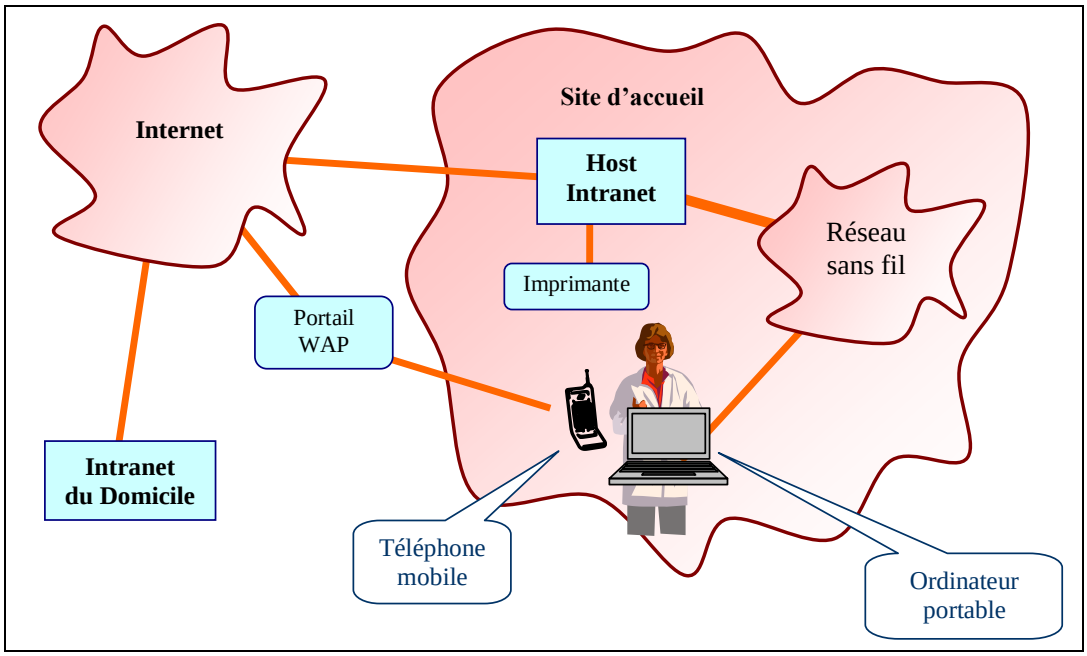

Figure 1.4. Exemple d'organisation pour systèmes mobiles

1.3 PROBLEMES

Actuellement, les applications distribuées apparaissent comme une dernière couche au dessous de laquelle il y a plusieurs autres. On en distingue trois telles qu'illustrées par la figure 1.5.

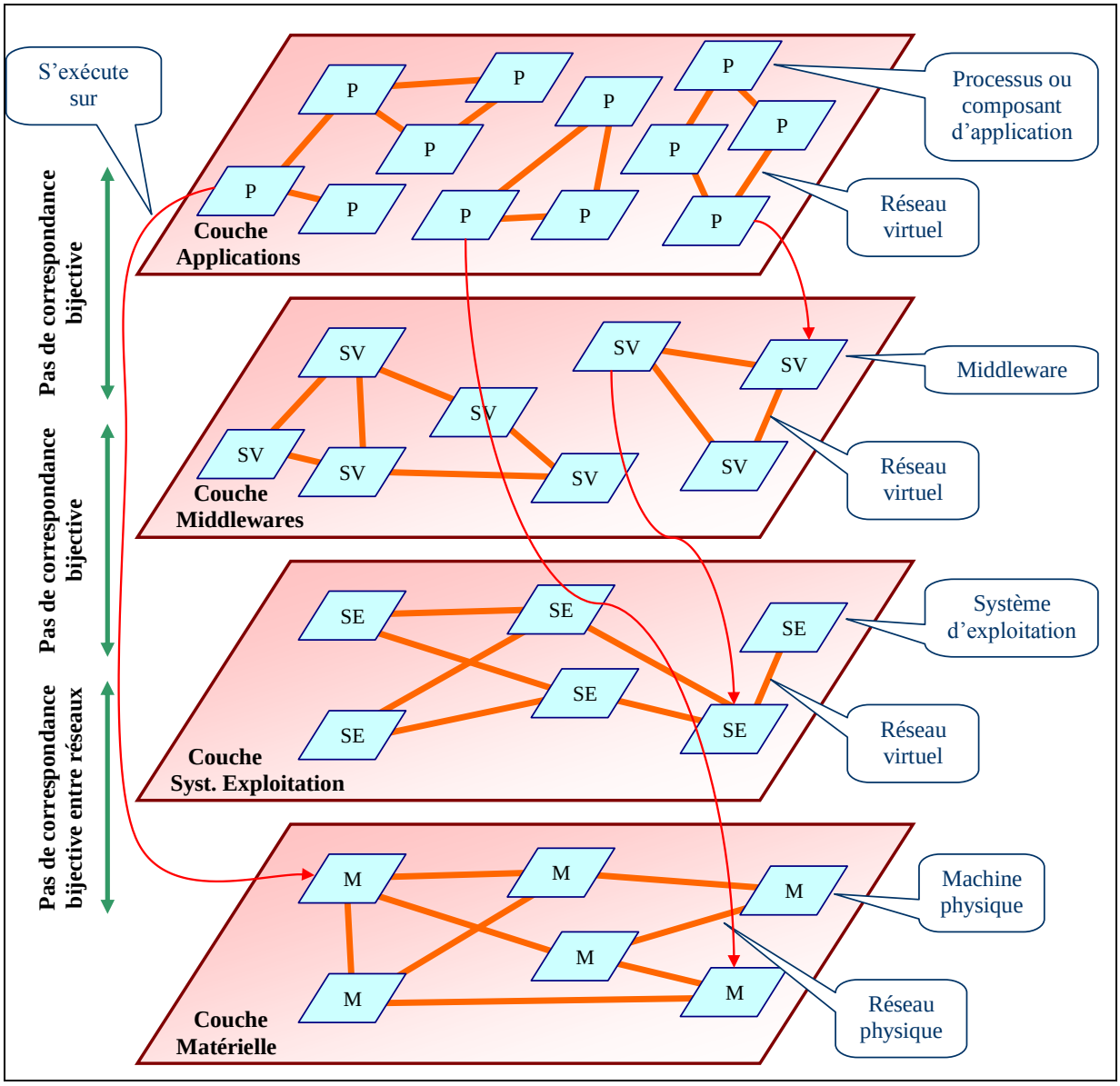

Figure 1.5. Structuration en couches des systèmes distribués

Dans chaque couche, il est possible d'avoir diverses entités reliées par un réseau qui est physique dans la couche matérielle et virtuel dans les autres couches. Chaque couche matérialise, à elle seule, un système distribué. Cependant, comme les entités de chaque couche s'exécutent en utilisant les services disponibles dans les couches inférieures, les applications distribuées sont donc mises en œuvre moyennant plusieurs systèmes distribués. La couche middleware est une couche spécifique pour les systèmes distribués qui est censée cacher les subtilités des couches inférieures et offrir aux

concepteurs des applications distribuées des services plus convenables de nature à réduire leur complexité.

Le schéma de la figure 1.5 laisse apparaître une complexité aussi bien au niveau de chaque couche mais également une complexité dans les relations qu'a une couche avec les autres. Cette situation engendre plusieurs problèmes auxquels la conception des systèmes distribués doit faire face, il s'agit de l'hétérogénéité, la concurrence, la sécurité, les pannes et l'absence d'informations globales. Nous discutons ces problèmes dans ce qui suit.

1.3.1 L'hétérogénéité (Heterogeneity)

Elle désigne les différences qui existent entre les différents composants d'un système distribué. Ces différences peuvent être au niveau de la couche matérielle (réseaux physiques et plateformes matérielles), de la couche systèmes d'exploitation (divers systèmes d'exploitation), de la couche middleware et de la couche application (utilisation de langages de programmation différents, etc.).

Dès lors en se retrouve devant une problématique : les composants d'un système distribués sont appelés à coopérer pour partager les ressources mais chaque composant peut avoir ses spécificités qui limitent ce partage ou le rendent difficile.

L'hétérogénéité au niveau matériel vient de la différence qui existe entre les ordinateurs aussi bien qu'entre les réseaux. Dans un même système distribué, il est possible de trouver un ordinateur personnel, une station de travail sophistiquée ou même un ordinateur multiprocesseur puissant. Les réseaux physiques présentent, eux aussi, beaucoup de différences : réseau de fibre optique, réseau sans fil, taille et structure des trames, etc.

La couche systèmes d'exploitation présente naturellement une hétérogénéité relativement importante malgré le fait qu'elle cache les subtilités de la couche matérielle. Ainsi, certains ordinateurs dans le système distribués peuvent être dotés d'un système d'exploitation Unix, d'autres utiliseront Windows, Mac OS ou Solaris, etc.

Au niveau middleware, il existe actuellement plusieurs possibilités. En effet, Microsoft propose sa plateforme .Net qui est censée faciliter la coopération entre les objets de diverses applications réalisées avec des langages de programmation différents. Corba est une spécification ouverte du consortium OMG qui a plusieurs réalisations pratiques qui offrent des moyens de partage et de coordination entre les composants des applications réparties. Les différents middleware sont appelés à coopérer malgré leur hétérogénéité pour faciliter la réalisation des applications distribuées.

Dans la couche application, l'hétérogénéité provient principalement de l'utilisation de langages de programmation différents, ce qui engendre des problèmes lors de la

manipulation des données, l'appel de procédures, etc. Par exemple, il peut y exister une différence dans le format des données où sur un ordinateur on utilise le stockage *Little Endian*, et sur un autre, un stockage *Big Endian*. De même pour la représentation des chaînes de caractères, les tableaux, les structures de données complexes (fichiers), etc.

1.3.2 Problème de la concurrence

Le problème de la concurrence provient de la manipulation simultanée des ressources par plusieurs programmes (processus). En effet, certaines ressources ne sont pas partageables et leur manipulation ne peut se faire que par un processus à la fois. C'est le cas des ressources physiques telles que l'imprimante mais aussi des ressources logiques telles que les fichiers, les tables des bases de données, etc.

Le problème de la concurrence se pose pour les systèmes distribués comme pour les systèmes centralisés (i.e. multiprocesseurs). La gestion de la concurrence dans les systèmes distribués nécessite des mécanismes et des outils tout comme les systèmes centralisés (moniteurs, sémaphore, etc.). Cependant, dans le cas des systèmes distribués, la gestion de la concurrence se base en partie sur la communication par messages et cette dernière introduit d'autres sources de difficultés, telles que la perte de messages ou leur corruption, d'en-t-il faut tenir compte.

1.3.3 Problème de la sécurité

Les problèmes de sécurité se posent pour tous les systèmes informatiques. Cependant, dans les systèmes distribués, la vulnérabilité se trouve accentuée par la répartition même. Ainsi, il est naturel dans le commerce électronique qu'une partie de l'application (un client) envoie des messages (par exemple numéro de la carte de crédit) à un autre (un serveur). Malheureusement le message peut être intercepté, lors de sa transmission, et son contenu divulgué, ce qui peut entraîner de conséquences graves aussi bien pour le client que pour le serveur.

D'autres problèmes se posent aussi, on cite :

- L'authentification : Comment le client (respectivement le serveur) peut-il être sûr de l'identité du serveur (respectivement le client) ? Par exemple, le client d'une banque a besoin de connaître l'identité de celle-ci avant d'ouvrir un compte et y déposer une somme d'argent, de même, le serveur doit connaître l'identité du client avant d'entreprendre une transaction quelconque sur son compte.
- L'attaque de services : Souvent les applications distribuées sont organisées selon le modèle client-serveur où le client invoque les services du serveur. Ce dernier dispose de ressources en nombre suffisant pour répondre à un nombre qui peut être important mais limité. L'attaque de service consiste à submerger le serveur par un nombre de requêtes qui excède ces capacités et ralentit considérablement sa cadence.

- Sécurité du code mobile : Le code mobile n'est autre qu'un programme qui migre d'une machine à l'autre dans un réseau dans le but d'accomplir une certaine tâche. A son arrivée, le code mobile utilise les ressources d'une machine pour accomplir sa tâche. Il peut donc présenter un risque s'il véhicule un virus ou un code malintentionné.

1.3.4 Problème des pannes

Les systèmes peuvent échouer dans l'accomplissement de leurs tâches suite à une faille matérielle ou logiciel qu'on appelle couramment *panne*. Les pannes peuvent conduire à des résultats erronés comme elles peuvent engendrer l'arrêt de toute ou partie de l'application distribuée.

Les pannes peuvent avoir lieu à différents niveaux de la structure en couches présentée dans la figure 1.5. Une panne au niveau matériel peut inclure la perte d'un message par le réseau de communication, des erreurs d'entrée/sortie, la corruption d'une donnée ou d'un message, etc. Ces pannes, au niveau matériel, se propagent aux niveaux supérieurs engendrant d'autres pannes de niveau supérieur, par exemple arrêt prématuré d'un processus.

Les pannes peuvent apparaîtrent aussi dans les différentes couches et se propager éventuellement aux autres couches. L'origine des pannes peut être une raison matérielle ou une raison logique liée à la conception des applications, des middlewares et des systèmes d'exploitation. Certaines peuvent avoir comme origine une faille dans la gestion de la concurrence ou la coordination en général. Les pannes d'ordre logiques sont innombrables du fait que les applications distribuées sont, par nature même, complexes. Par exemple, l'omission, dans le programme d'un processus, de la libération d'une ressource peut conduire à une situation d'interblocage. L'absence de l'exclusion mutuelle pour l'accès à une ressource critique, par exemple un tampon, peut avoir des répercussions importantes sur l'ensemble de l'application distribuée.

1.3.5 Absence des informations globales

Dans un système centralisé, une application se compose de plusieurs processus qui coopèrent pour réaliser un objectif commun. A tout moment, il est possible d'avoir des informations sur l'état de chaque processus et sur l'état des différentes ressources. Ainsi, chacun des processus peut avoir une vision globale de l'application et de son environnement. La présence de cette vision globale facilite grandement la mise au point des mécanismes de coordination des processus dont, par exemple, la gestion des ressources.

La vision globale dans les applications centralisées est due au fait que les processus partage une mémoire commune qui renferme les informations sur les ressources et les processus.

Dans le cas des systèmes distribués, la mémoire commune est inexistante. Chaque processus se trouve dans un environnement différent (système d'exploitation différent) et son état est une information interne à ce dernier. De même, les informations sur les ressources partagées ne sont pas accessibles en temps réel. De ces faits, la coordination d'un ensemble de processus distribués nécessite le recours à la communication par échange de messages. Malheureusement, un message parvient toujours à sa destination avec un certain retard et par conséquent l'information qu'il véhicule ne reflète pas toujours l'état d'un processus ou d'une ressource, car ces derniers peuvent changer d'état durant la transmission du message.

Il est impossible d'avoir une vision globale instantanée de l'état d'un système distribué et cette situation rend la coordination dans ces derniers un véritable challenge.

1.4 DEFIS ET OBJECTIFS

Comme souligné dans la définition des systèmes distribués, l'objectif essentiel est le partage des ressources. Le concepteur d'une application répartie doit, en plus du développement des programmes assurant les fonctionnalités de celle-ci, tenir compte des différents problèmes cités précédemment et prévoir des mécanismes pour y faire face de telle sorte à faire apparaître le système distribué comme un tout cohérent.

Ainsi, à la complexité inhérente aux applications réparties, s'ajoute une complexité due à la distribution, ce qui constitue un défi considérable pour les concepteurs.

Pour faciliter le travail des concepteurs, les couches middlewares ont vu le jour. Ces dernières offrent des services plus convenables à utiliser. L'idée de simplifier le développement des applications en ajoutant des couches qui prennent en charges certains aspects liés à la répartition n'est pas nouvelle. En effet, les systèmes d'exploitation eux mêmes comprennent des services spécifiques à la répartition. Cependant, les disparités entre les divers systèmes d'exploitations et l'impossibilité d'arriver à un consensus ont conduit à la création des couches middlewares.

Faire face aux problèmes des systèmes distribués signifie apporter une solution à chacun soit au niveau du middleware (voire au niveau des couches inférieures) soit au niveau de l'application elle-même. Résoudre ces problèmes au niveau des middlewares vise en premier lieu à les masquer pour le concepteur de l'application et par conséquent pour l'utilisateur aussi. Si par contre, les problèmes sont résolus au niveau des applications, on les masque pour les utilisateurs mais ce sont les concepteurs de ces applications qui doivent les prendre en charge.

Quelque soit le niveau considéré, les solutions apportées doivent garantir des propriétés fondamentales suivantes : l'interopérabilité, l'ouverture, l'invariance à l'échelle (scalability), la gestion de la sécurité, l'efficacité et la disponibilité, le maintien de la

consistance des ressource, la transparence (flexibilité), la gestion des transactions réparties, la tolérance aux fautes et la gestion des situations d'exception (détection des pannes et reprise) etc.

Ces propriétés constituent de véritables défis pour les concepteurs de systèmes distribués. Nous les discutons dans les paragraphes suivants.

1.4.1 L'interopérabilité

L'interopérabilité est une caractéristique importante qui désigne la capacité à rendre compatibles deux systèmes quelconques. A son tour, la compatibilité est la capacité qu'ont deux systèmes à communiquer sans ambiguïté. L'interopérabilité nécessite que les informations relatives aux parties de communication des systèmes considérés soient disponibles sous la forme de standards ouverts.

L'interopérabilité vise à réduire le problème de l'hétérogénéité. Le masquage de celle-ci passe en premier lieu par l'utilisation d'un protocole unique de communication (e.g. cas de TCP/IP pour Internet). Pour établir un échange de message, il faut utiliser des standards qui cachent les différences entre les différentes plateformes.

Il existe actuellement deux approches principales de standardisation pour masquer l'hétérogénéité : les middlewares et les machines virtuelles.

Un middleware est une couche logicielle qui masque l'hétérogénéité en offrant aux programmeurs d'une application un modèle de programmation plus convenable. Concrètement un middleware est représenté par des processus et des ressources d'un ensemble d'ordinateurs, qui interagissent les uns avec les autres (communiquent et partagent des ressources).

Un middleware, tel que CORBA ou DCOM, offre des services qui peuvent être utilisés pour construire des composants logiciels pouvant fonctionner ensemble dans un système distribué.

Les middlewares améliorent la communication en offrant des abstractions telles que

- Le RMI (Remote Method Invocation) : C'est la possibilité, pour un objet, d'invoquer la méthode d'un autre objet situé sur une plateforme distante.
- La notification d'événement : Les événements représentent un moyen efficace de propagation d'information d'une plateforme vers une autre ou plusieurs autres plateformes ou composants d'une application répartie.
- Communication entre groupes de processus
- Gestion de la duplication des données partagées
- Transmission de données multimédia en temps réel

Les machines virtuelles permettent de supporter le code mobile. Ce dernier désigne la possibilité de transfert de code d'une machine source à une machine destination et son exécution sur cette dernière. En effet, si les plateformes sont différentes, le code produit pour l'une ne peut fonctionner sur l'autre. Pour éviter ce problème, le code mobile est généré d'un langage source pour une machine virtuelle donnée (e.g. Machine virtuelle de Java :JVM).

Chaque plateforme doit disposer d'une couche logicielle qui implémente la machine virtuelle. Les machines virtuelles représentent une généralisation des middlewares dans le sens où elles n'offrent pas seulement quelques services communs mais offrent les mêmes services.

1.4.2 L'ouverture (Openness)

Elle désigne la possibilité d'ajout, de suppression ou de modification des ressources et services dans un système distribué.

L'ouverture nécessite que les interfaces logicielles soient documentées est accessibles aux développeurs d'applications (i.e. publiées publiquement).

Le défi que pose l'ouverture vient du fait que les composants d'un système réparti ont des origines diverses. Dans ce cadre, on qualifie un système d'*ouvert* s'il supporte, sans difficultés :

- L'ajout d'ordinateurs au niveau de la couche matérielle
- L'ajout de nouveaux services au niveau application, middleware et système d'exploitation
- La réimplémentation des services anciens

Les systèmes ouverts présentent implicitement une indépendance vis-à-vis des constructeurs des plateformes et fournisseurs de produits logiciels.

1.4.3 L'invariance à l'échelle (Scalability), efficacité et disponibilité

Un système est dit invariant à l'échelle (scalable system) s'il reste performant et d'un coût raisonnable lors d'une augmentation importante du nombre des utilisateurs et de ressources gérées. Par exemple, Internet a connu une évolution exponentielle qui a fait passer le nombre d'ordinateurs de 188 en 1979 (début d'Internet) à 56 millions en 1999. De ce fait, il peut être considéré comme un système invariant au changement de l'échelle des utilisateurs et des ressources.

Avoir un système invariant à l'échelle signifie :

- L'ajout d'utilisateurs et de ressources est toujours possible

- Le coût d'ajout est proportionnel au nombre d'utilisateurs ou ressources ajoutés
- Maintien de bonnes performances ou baisse raisonnable suite aux ajouts

1.4.4 Gestion de la sécurité

Résoudre le problème de sécurité revient à assurer :

- La confidentialité des informations vis-à-vis des personnes non autorisées à la connaître
- L'intégrité vis-à-vis des altérations ou corruptions (i.e. modifications malintentionnées)
- La disponibilité du système (i.e. pas d'interférence entre composants qui engendre un blocage du système ou détérioration des performances)

Dans un système réparti, l'échange de message nécessite leur protection et l'authentification de leur émetteur (e.g. cas du commerce électronique). La cryptographie et les mots de passe sont très utilisés actuellement. Cependant, malgré leur efficacité, ils ne permettent pas d'éradiquer complètement les problèmes épineux tels que l'attaque des services et la sécurité du code mobile.

1.4.5 Gestion de la concurrence

Dans les systèmes répartis, une ressource critique peut être utilisée simultanément par plusieurs processus physiquement répartis. Une manière simple de gérer cette concurrence consiste à créer un processus dit *allocateur de la ressource* qui accepte les demandes des processus dits clients (i.e. qui souhaitent utiliser la ressource) puis autorise un processus à la fois à utiliser la ressource. Cette solution est contraignante car le processus allocateur réduit les performances et peut tomber en panne.

Les applications réparties actuelles autorisent l'exécution de plusieurs services en concurrence (e.g. l'accès à une base de données). Chaque demande est prise en compte par un processus simple, dit thread, et la gestion de la concurrence fait appel aux mécanismes de synchronisation classiques tels que les sémaphores.

D'autres approches plus complexes existent, il s'agit d'établir une coordination entre les processus clients (dits alors processus pairs), à chaque demande, de telle sorte que l'un d'entre eux puisse accéder à la ressource au bout d'un temps fini.

Notons que la gestion de la concurrence dans un système distribué implique sa gestion aussi dans le cas d'un système centralisé car les parties impliquées dans un système distribué peuvent être des ordinateurs multitâches.

1.4.6 La transparence

En général, la transparence d'un mécanisme ou d'un concept signifie son existence et son utilisation implicite sans que l'utilisateur d'une application (qui peut être le concepteur de l'application) ne s'en rende compte. En d'autres termes, un concept est transparent dans un système si ce dernier se comporte, vu de l'utilisateur, comme si le concept n'y existait pas. Dans les systèmes distribués, la transparence signifie que la répartition des composants reste cachée pour l'utilisateur. Ce dernier perçoit le système distribué comme un tout et non comme une collection de composants indépendants. Dans ce contexte, on distingue huit formes de transparence :

- Transparence d'accès : Il s'agit d'utiliser les mêmes opérations pour l'accès aux ressources distantes que pour les ressources locales.
- Transparence à la localisation : Il s'agit de permettre une transparence d'accès aux ressources sans une connaissance préalable de leur localisation.
- Transparence à la concurrence : Il s'agit de cacher à l'utilisateur la gestion de la concurrence en évitant principalement les interférences.
- Transparence à la duplication : Dans le but d'accroître la fiabilité (implémenter d'autres formes de transparence) et améliorer les performances, on a souvent recours à la duplication de certaines ressources (e.g. fichiers de base de données). La gestion de cette duplication ne doit pas être perceptible par l'utilisateur.
- Transparence aux pannes : Il s'agit de permettre aux applications des utilisateurs d'achever leurs exécutions malgré les pannes qui peuvent affecter les composants d'un système (composants physiques ou logiques).
- Transparence à la mobilité : Il s'agit de permettre la migration des ressources et des clients à l'intérieur d'un système sans influencer le déroulement des applications.
- Transparence à la reconfiguration : Dans le but d'améliorer les performances en fonction de la charge (i.e. les demandes de services des clients), il devrait être possible de reconfigurer un système sans que cela ne soit perceptible par l'utilisateur. Par exemple, les sites dits miroirs permettent d'améliorer les performances de l'accès à travers Internet, les déroutements vers ces sites doivent être non perceptible pour l'utilisateur.
- Transparence à la modification de l'échelle (Scaling transparency) : C'est la possibilité d'une extension importante d'un système sans influence notable sur les performances des applications.

Remarque : Dans certains cas la transparence n'est pas souhaitable. Par exemple la duplication d'une imprimante améliore les performances, cependant l'utilisateur doit toujours avoir la possibilité de spécifier explicitement sur quelle imprimante il souhaite

imprimer ses documents. Cet aspect est particulièrement important si les imprimantes n'ont pas les mêmes caractéristiques (qualités) ou ne sont pas situées dans un même endroit.

1.4.7 Gestion des situations d'exception

Assurer la gestion des situations d'exception ou pannes fait intervenir plusieurs notions :

- Détection : Certaines pannes ou erreurs peuvent être détectées facilement (cas des erreurs de transmission), par contre, d'autres pannes ne peuvent que être suspectées. Par exemple, la panne d'un processus serveur ou allocateur d'une ressource ne peut être détectée avec certitude. On ne peut conclure à une panne alors que le serveur n'est peut être que surchargé ou que les messages sont perdus. D'un autre coté, on peut attendre indéfiniment alors que le serveur est en panne. C'est un dilemme difficile à résoudre.
- Masquage : Il s'agit de rendre les pannes autant que possible transparentes. Pour cela, différentes solutions peuvent être imaginées. Par exemple :
  - Retransmission des messages qui ne parviennent pas à destination
  - Duplication des ressources importantes pour les maintenir accessibles en cas de problèmes (cas de corruption d'un fichier)
- Tolérance : Certains composants doivent être programmés (voire conçus) pour tolérer des pannes ou transmettre la détection des pannes aux utilisateurs (e.g. panne non masquable). Un composant est dit tolérant, s'il profite de/gère la redondance de ressource.
- Reprise : La reprise après pannes désigne la possibilité de remise du système (principalement les données qu'il manipule) dans un état cohérent après la détection d'une panne.