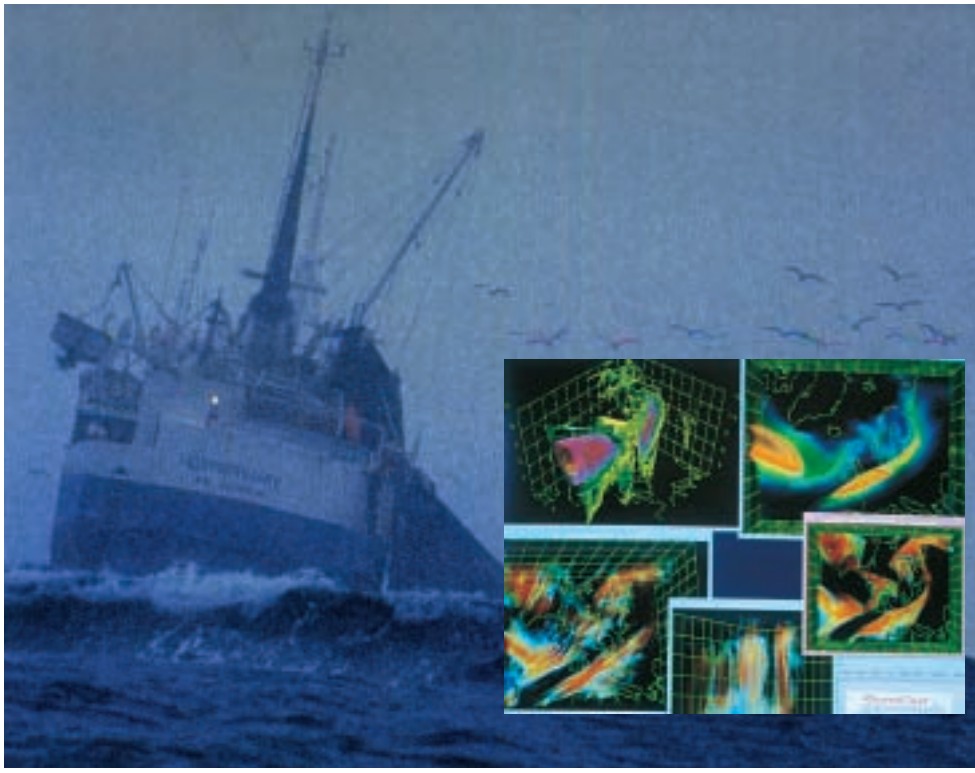
Des passerelles fiables

Même une brève défaillance des systèmes répartis est grave quand les programmes utilisés doivent fonctionner en permanence. Ainsi, les réseaux qui gèrent le trafic aérien ou les opérations bancaires doivent être extrêmement fiables et constamment remis à jour : un message tel que «l'ordinateur central ne répond pas» ou l'affichage d'informations périmées sur le trajet d'un avion en vol ou la cote de valeurs boursières risqueraient de provoquer des accidents ou des désastres financiers. Or les systèmes informatiques répartis se multiplient ; leur fiabilité devient vitale tant pour les entreprises que pour les particuliers. On vante aujourd'hui les avantages considérables des autoroutes de l'information, mais on n'obtiendra les avantages rêvés que si les liaisons entre les ordinateurs sont étudiées de près. Depuis la fin des années 1970, de nombreux informaticiens recherchent des programmes qui rendraient les réseaux plus sûrs et plus fiables, bref plus «robustes».

Pourquoi les réseaux tombent-ils en panne? Habituellement, un problème isolé en un site quelconque se propage dans le réseau, ce qui provoque l'interruption en chaîne des programmes, ordinateur après ordinateur. Il serait vain de renforcer les composants individuels, par exemple en introduisant des ordinateurs et des disques spécialement conçus pour tolérer les erreurs, car on ne parerait pas aux incidents imprévisibles tels qu'une fuite d'eau dans la salle informatique ou une panne de courant ; de surcroît, les communications peuvent être coupées par inadvertance ou par sabotage. Même si les ingénieurs et les programmeurs améliorent sans cesse les matériels et les logiciels, les ordinateurs ne sont jamais totalement fiables.

En outre, des ordinateurs fiables interconnectés et des programmes sans erreur ne conduisent pas nécessairement à des systèmes répartis robustes ; ils constituent seulement des réseaux qui fonctionnent correctement dans la plupart des conditions. Les programmes de gestion du courrier électronique, les tribunes de discussion et le *Web* ont été réalisés à partir de composants individuellement très fiables. Néanmoins, ces systèmes se bloquent fréquemment lorsqu'un événement inattendu survient dans l'un des composants du système. Le système tombe en panne lorsqu'une machine ou une ligne de transmission est surchargée. D'autres formes de protection sont donc nécessaires.



2. UN RÉSEAU DE SURVEILLANCE MÉTÉOROLOGIQUE alerte les pêcheurs norvégiens en cas de tempêtes dangereuses *(à gauche)* ou de marée noire *(à droite)*. Le système informatique *StormCast* relie des caméras éloignées, des stations météorologiques et des

Depuis 20 ans, les programmeurs fiabilisent les réseaux en mettant au point des logiciels qui tolèrent les erreurs : ces programmes permettent aux systèmes informatiques de quitter une tâche quand un blocage survient. Plus généralement, on cherche à éliminer les chaînes de dépendance internes qui subordonnent le fonctionnement d'un système réparti au fonctionnement de tous ses composants. Les systèmes qui en résultent continuent à fonctionner même lorsque certains sites sont déconnectés : le système change sa configuration de manière à contourner les serveurs indisponibles.

Sauvés par les sauvegardes

Les systèmes qui possèdent cette aptitude sont des systèmes répartis à haute disponibilité. En dupliquant en permanence les informations critiques et en distribuant les multiples copies de sauvegarde à leurs ordinateurs individuels, ces systèmes s'adaptent à des aléas de fonctionnement tels qu'un lecteur de disque défectueux en un site, une surcharge dans un autre, une connexion coupée, etc. Tant que les défaillances surviennent à un rythme qui laisse aux programmes le temps de s'adapter, ces systèmes peuvent répondre aux requêtes en extrayant d'un site opérationnel une copie du fichier vital ou la copie d'un programme utilisé en ligne. Le système tout entier reste ainsi opérationnel et, dans l'idéal, fournit un service ininterrompu aux utilisateurs connectés.

On obtient, de manière simple, un système réparti à haute disponibilité en associant une machine principale et une machine de secours. Lorsque la machine principale tombe en panne, la machine de secours prend la relève. La commutation d'un système à l'autre est facile lorsque les données ne changent jamais. En revanche, la commutation est difficile lorsque les données ou les fichiers changent alors que le système fonctionne. De plus, dans les grands réseaux, on distingue difficilement les systèmes en panne des systèmes qui n'ont que des difficultés avec les transmissions.

Supposons que, lors de la mise à jour des informations du serveur principal et du serveur de secours, l'un des deux ne réponde plus. Si le problème

satellites pour fournir des données mises à jour. On accède à *StormCast* par le *World Wide Web* (la «toile d'araignée mondiale» véhiculée par le réseau *Internet*) à l'adresse http://www.cs.uit.no/

ne provient que des lignes de communication, la mise à jour finira par se faire. En revanche, si le serveur qui ne répond plus est hors service, l'ordinateur qui fait la mise à jour attendra indéfiniment ; pendant ce temps, le système sera indisponible. Si l'on se contente d'ordonner à l'ordinateur de cesser l'attente et de ne mettre à jour que le serveur qui fonctionne, le système principal et celui de secours ne seront plus identiques : des erreurs surviendront si le système tente d'utiliser le serveur périmé.

Le marché financier NASDAQ résout ce problème grâce à deux serveurs centraux d'échanges boursiers. Pour éviter toute confusion, un seul des deux serveurs est actif à un instant donné. Les administrateurs du NASDAQ décident eux-mêmes quand basculer sur le serveur de remplacement. Malheureusement, très peu de systèmes distribués peuvent compter sur la sagesse d'un opérateur humain pour détecter les pannes et décider le changement d'un serveur ; les programmeurs doivent généralement automatiser cette décision, afin que la transition survienne sans heurts.

En outre, les systèmes répartis à haute disponibilité gèrent souvent un grand nombre de serveurs et de programmes. Pour ce faire, les systèmes tiennent à jour la liste de leurs membres et gardent ainsi la trace de chaque programme et de son état d'activité. Un programme qui, pour quelque raison que ce soit, ne répond pas est jugé défectueux. En reconnaissant une défaillance en un site, le système change alors sa configuration et redistribue le travail vers les sites opérationnels.

Le système NASDAQ illustre également un second problème de fiabilité des systèmes distribués. Le retard de deux heures dans le début des échanges financiers, en 1994, aurait été évité si les opérateurs avaient immédiatement enclenché le système de secours. Ils choisirent pourtant d'attendre, car ils craignaient qu'une erreur logicielle n'ait causé le mauvais fonctionnement du système principal. Dans ce cas, la panne se serait produite également sur le système de secours, comme ce fut le cas en 1994 pour la panne du système de la compagnie de téléphone *AT&T*, mentionnée auparavant. Comme il est impossible de garantir qu'un logiciel soit exempt d'erreurs, on doit trouver des moyens de réduire le risque que les versions de secours d'un serveur critique ne tombent en panne à leur tour après la défaillance du serveur principal.

À cette fin, les programmeurs ont inventé la duplication active : un programme fait des copies de secours des programmes et des fichiers vitaux en utilisant des «groupes de processus». Un groupe de processus est un ensemble de programmes qui coopèrent. Un système réparti peut comporter plusieurs groupes de processus, et les programmes peuvent appartenir à plusieurs de ces groupes. Chaque groupe est identifié par un nom semblable à un nom de fichier et possède sa propre liste actualisée de participants (l'ensemble des programmes qui coopèrent). En outre, le groupe de processus permet d'envoyer des messages à ses membres. Cette fonction de transmission des messages garantit que chaque membre du groupe reçoit chaque message dans le même ordre, même si l'émetteur tombe en panne pendant la transmission, ce qui est vital pour de nombreuses applications.

Lorsqu'un programme particulier est vital pour maintenir la disponibilité du système, ce dernier crée un groupe de programmes qui sont, chacun, une copie de l'original. Pour garder à jour les données manipulées par le programme dupliqué, le système envoie un message au groupe de programme. Chaque membre réagit en mettant à jour sa copie du programme. Comme tous les programmes «voient» les mêmes mises à jour dans le même ordre, ils restent dans des états cohérents.

La duplication active permet à un système de tolérer les erreurs, car n'importe quel membre du groupe peut traiter n'importe quelle requête : quand une machine tombe en panne, la tâche est confiée à un site opérationnel. En outre, quand une requête ne modifie pas les données, elle est satisfaite par un site unique au lieu de mettre tout le système à contribution. De cette façon, de multiples tâches sont menées en même temps par différents programmes, ce qui accélère l'application en utilisant le traitement parallèle.

Naturellement, si tous les membres d'un groupe de processus traitaient de manière erronée un message, ils tomberaient tous en panne simultanément. Toutefois, les erreurs de ce genre sont exceptionnelles, car les programmeurs ont observé que les erreurs qui passent le plus facilement inaperçues lors des tests du logiciel sont plutôt celles où intervient l'ordre de réception des données. Ces erreurs sont provoquées uni-