Service de Détection de Pannes avec SNMP

Matthias Wiesmann
JAIST, 旭台 1-1 能美市 石川県
Tel. : +81 761 51 1254 - Fax. : +81 761 51 1149
E-mail : wiesmann@jaist.ac.jp

Résumé :
La détection de pannes est un aspect important de la programmation distribuée. Dans un réseau informatique, il est en effet difficile de déterminer si un nœud est non fonctionnel ou simplement lent. Or, la distinction est importante pour une application distribuée : dans le premier cas une reconfiguration est nécessaire, dans le second, une simple attente suffit souvent. Implémenter une détection de pannes efficace est un problème difficile, ce qui fait qu'en général, les programmes utilisent des approches simplistes comme des temps limites fixés.

Pour contrer ce problème, la notion de service de détection de panne a été proposée depuis plusieurs années. Un tel service offre de nombreux avantages : il permet de n'avoir qu'une seule implémentation partagée par plusieurs applications, cette implémentation peut-être changée sans affecter les applications, et le service peut utiliser des techniques complexes sans compliquer les applications.

Plusieurs services de détection de pannes ont été proposés, mais aucun n'est utilisé en dehors de sa niche d'origine. Une des raison à cela est le fait qu'il n'existe aucune interface standard pour un tel service. Nous présentons un service de détection de panne basé sur le standard SNMP (Simple Network Management Protocol). Ce standard est très largement utilisé pour la surveillance de l'infrastructure réseau ou de serveurs. Notre service utilise SNMP pour les communications et les interfaces avec les applications. De cette manière, il peut être interfacé en utilisant des technologies et des outils existants.

Abstract :
Failure detection is an important issue of distributed programming. In a computer network, distinguishing a failed node from a slow one is difficult. Yet the difference is important for distributed applications: in the first case a reconfiguration is needed, in the second case, waiting is sufficient. Implementing efficient failure detection is difficult. Because of this, many applications use simplistic approaches like fixed timeouts.

To address these issues, failure detection services have been proposed for some years. A service offers many advantages: a single implementation is shared by multiple applications, this implementation can be changed without rebuilding the application and complex failure detection techniques can be used without weighting down or increasing the complexity of the application.

Many failure detection services have been proposed. But none is used outside of a small application niche. One of the reasons for this is that there is no standard for such a service. We present a failure detection service based on the SNMP (Simple Network Management Protocol) standard. SNMP is widely used for monitoring network infrastructure and servers. Our failure detection service uses SNMP for communication and application interfaces. This way, it can be interfaced using standard technologies and existing tools.

Introduction :
Une définition d'un système réparti a été donnée par Leslie Lamport : *«Vous êtes en présence d'un système réparti si l'arrêt d'un ordinateur dont vous n'avez jamais entendu parler vous empêche de*

travailler». Elle illustre un des problèmes centraux dans les systèmes distribués: détecter et gérer les défaillances. Pour pouvoir tolérer des fautes, un mécanisme pour détecter les pannes de machines distantes est nécessaire. Au fur et à mesure que la taille et la complexité d'un système croissent, le besoin d'un système de détection de pannes devient une nécessité.

Si la détection de pannes est un aspect important des systèmes répartis, la problématique est plus générale. Elle touche la gestion de réseau et de grappes de machines, le déploiement d'applications et l'informatique distribuée au sens large. S'il existe des solutions pour surveiller processus et nœuds, elles sont en général restreintes à des tâches de surveillance et de gestion. Au niveau applicatif, la détection de pannes se fait typiquement avec des *timeouts* (temps limites).

L'approche par *timeout* n'est pas adaptée à la détection de pannes. Si l'on considère par exemple un client Web qui envoie une requête HTTP, un *timeout* durant la connexion peut signifier deux choses : le serveur est en panne, ou bien il est lent. L'utilisateur n'a aucune manière de distinguer ces deux cas à moins d'exécuter à nouveau la requête, ce qui n'est ni pratique, ni fiable. En fait, si le serveur est surchargé, cette nouvelle requête va simplement ajouter à la charge. Fixer le bon *timeout* est une tâche difficile, s'il est trop court, un serveur lent sera considéré comme défaillant, s'il est trop long, le client attendra inutilement la réponse d'un serveur en panne. Comme il n'existe pas de service de détection de pannes, les *timeouts* restent malheureusement la seule solution dans de nombreux cas.

Un service de détection de pannes a pour but de remédier à ce problème. Les applications utilisent ce système pour décider si un processus distant est en panne, et, partant de là, elles se reconfigurent en cas de défaillances. Ce service peut-être utilisé directement par les applications, mais peut être utilisé par un *middleware* comme CORBA ou une infrastructure GRID. Finalement, ce service peut-être utilisé pour implémenter des détecteurs de fautes qui fournissent des propriétés formelles comme $\diamond S$ ou Ω[2].

Un service de détection offre de nombreux avantages. Premièrement il permet d'extraire une fonctionnalité complexe et par là de simplifier les applications. Deuxièmement l'implémentation, l'algorithme et le modèle sous-jacent peuvent être changés sans nécessiter de modification dans l'application. Troisièmement, ce service peut implémenter des techniques complexes qui ne seraient pas acceptables dans des applications – typiquement des approches adaptatives qui intègrent plusieurs sources d'information. Quatrièmement un service peut consolider les informations de *monitoring* de plusieurs applications en un seul flot, ce qui permet l'économie de messages.

Si l'idée d'un service de détection de fautes n'est pas nouvelle, aucun des services proposés n'est utilisé en dehors de sa communauté d'origine. Souvent, les services ont été conçus pour servir à l'intérieur d'un projet ou d'un *toolkit* particulier. De ce fait, l'interopérabilité et l'intégration avec des standards existants n'a pas été une priorité. Un service sans interfaces standardisées a peu de chances d'être accepté à large échelle [1].

Le système SNMP-FD offre un service de détection de pannes qui cherche à remédier à ce défaut. Le service est basé sur un standard établi depuis plus de vingt ans dans le domaine de la gestion de l'infrastructure réseau et de la surveillance de serveurs critiques. L'utilisation du protocole SNMP nous offre plusieurs avantages : le protocole SNMP est un protocole léger, il nous permet d'interopérer avec les équipements réseau et d'autres services de gestion et de monitoring, enfin l'utilisation d'un standard existant nous évite d'en définir un nouveau, qui aurait de la peine à être accepté.

Middleware :
L'approche habituelle pour assurer l'interopérabilité dans les systèmes répartis est d'utiliser un middleware. Ces outils standardisent les interactions sur le réseau et le format des données. Plusieurs *middlewares* ont été envisagés pour ce projet, notamment CORBA et SOAP, mais SNMP est

clairement l'outil le plus adapté : le protocole a été conçu pour des tâches de surveillance, permet de communiquer avec les équipements réseau, et une bonne partie des interfaces pour la surveillance de processus est déjà définie. De plus, étant conçu pour être embarqué sur du matériel réseau, c'est un protocole simple qui utilise des messages courts. Les autres middleware offrent des capacités et des abstractions bien plus avancées, mais qui, dans notre cas, ne sont pas utiles.

SNMP :
SNMP est un standard défini par l'IETF dans les années quatre-vingt-dix pour la gestion d'équipements réseau. De nos jours la grande majorité des équipements réseau (routeurs, passerelles, *switches*) ainsi que de nombreux périphériques connectés à travers le réseau (imprimantes), comportent un agent SNMP et peuvent donc êtres gérés en utilisant des outils standard.

L'agent est l'entité logique responsable de surveiller un équipement et gérer les interactions SNMP. L'agent maintient une base de donnée, la *Management Information Base* (MIB), qui reflète l'état de l'agent et de l'équipement. La MIB est structurée en un arbre logique : chaque sous-arbre contient des données pertinentes à un domaine précis. Dans sa forme la plus simple, cette MIB ne contient qu'un sous-arbre avec des informations administratives (emplacement, responsable, etc.). De nombreux sous arbres ont été standardisé : allant de configuration et l'état des interfaces réseau jusqu'à l'état des bacs à papier d'une imprimante. Tous les éléments de la MIB peuvent être lus et dans certains cas, écrits. Un sous-arbre très important contient les tables de notification. Des processus distants peuvent y enregistrer leurs adresses afin de recevoir des messages asynchrones lors de changement d'états de l'équipement. Ces messages sont nommés *trappes* (*trap*). Ce mécanisme évite de devoir régulièrement interroger l'équipement pour savoir si son état a changé.

SNMP-FD :
Le service de détection de fautes SNMP-FD est conçu d'entrée pour être utilisé dans des contextes multiples et à travers une interface standard. Il est intégralement construit sur le standard SNMP. Il utilise ce standard deux manières.

Premièrement, les interfaces du service sont exposées suivant le standard SNMP : l'état du service est accessible via la MIB et les suspicions de pannes sont délivrées par le biais de trappes. Les informations propres à un service de détection de pannes sont définies dans des sous-arbres spéciaux, mais une grande partie de l'information (état des processus, enregistrement des entités désirant recevoir des notifications) est stockée dans des sous-arbres standard. Notre outil est donc compatible avec les outils de monitoring et d'administration réseau existants.

Deuxièmement, le service est construit en utilisant les mécanismes offerts par le protocole SNMP. Les messages entre les différents nœuds sont des messages SNMP. En particulier, un aspect important de l'implémentation d'un service de détection de pannes est l'envoi de messages de *heartbeat*, ces messages sont échangés périodiquement entre les nœuds afin de vérifier que chaque nœud n'est pas défaillant. Ces messages sont implémentés au moyen de trappes SNMP. L'avantage de cette approche c'est que les trappes SNMP ont été conçues pour minimiser la charge réseau : une trappe est transmise grâce à un seul message UDP.

Le service SNMP-FD est conçu de manière à permettre en son sein l'implémentation des diverses politiques de détection de pannes décrites dans la littérature. Toutes ces techniques de détection partagent une infrastructure commune fournie par notre système. De plus, l'intégration avec le protocole SNMP permet d'utiliser non seulement des messages de *heartbeat,* mais aussi des informations extraites de l'équipement réseau, ces informations aident à améliorer la qualité du service [3]. Par exemple, si une machine tombe en panne, son interface réseau sera désactivée. Cet événement sera détecté par le *switch* auquel la machine est connectée et annoncé au moyen d'une trappe. L'utilisation de ces informations avancées permet d'augmenter la qualité du service de détection sans augmenter la fréquence d'envoi des messages de *heartbeat,* en effet, ceux-ci tendent à surcharger le réseau s'ils sont envoyés à une trop grande fréquence.

Architecture :

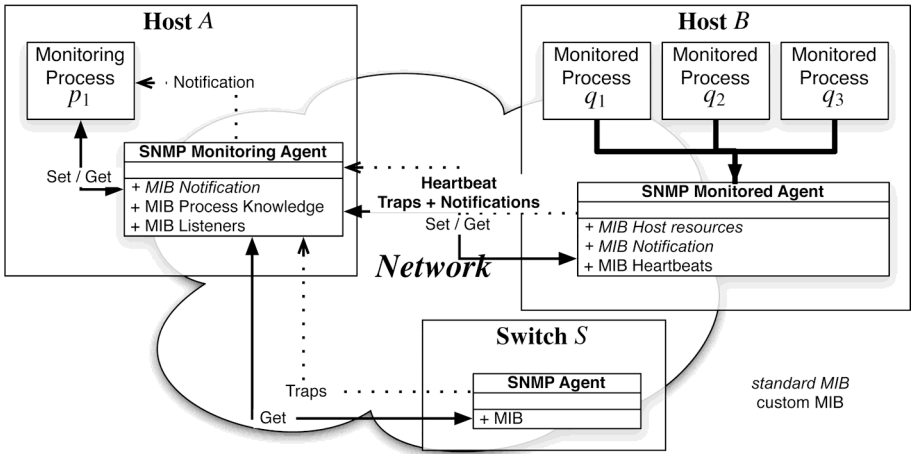

Figure 1 : Architecture du Service SNMP-FD

La Figure 1 illustre l'architecture du Système SNMP-FD. Le processus p_1 sur le nœud A surveille trois processus q_1, q_2 et q_3 sur le nœud B. Le nœud B contient un processus démon qui implémente le répondeur SNMP et surveille les processus locaux. Le switch S connecte les nœuds A et B, en cas de défaillance du lien S-B une trappe est envoyée au nœud A. Les trappes qui parviennent au nœud A ont des significations différentes, une trappe de *heartbeat* du nœud B indique une *absence* de défaillance de B, alors qu'une trappe en provenance du *switch* S indique la *présence* d'une défaillance du nœud B. Le nœud A contient un démon SNMP local. Celui-ci est responsable d'interpréter ces différentes informations et de transmettre l'interprétation en termes de suspicions de pannes à l'application cliente (le processus p_1).

Implémentation

Un premier prototype a été réalisé durant des projets d'étudiants à l'École Polytechnique Fédérale de Lausanne [4]. Ceux-ci ont démontré la faisabilité de l'approche, et de bonnes performances dans un réseau local [5] ainsi qu'une analyse des données pouvant être extraites des équipements réseaux [6].

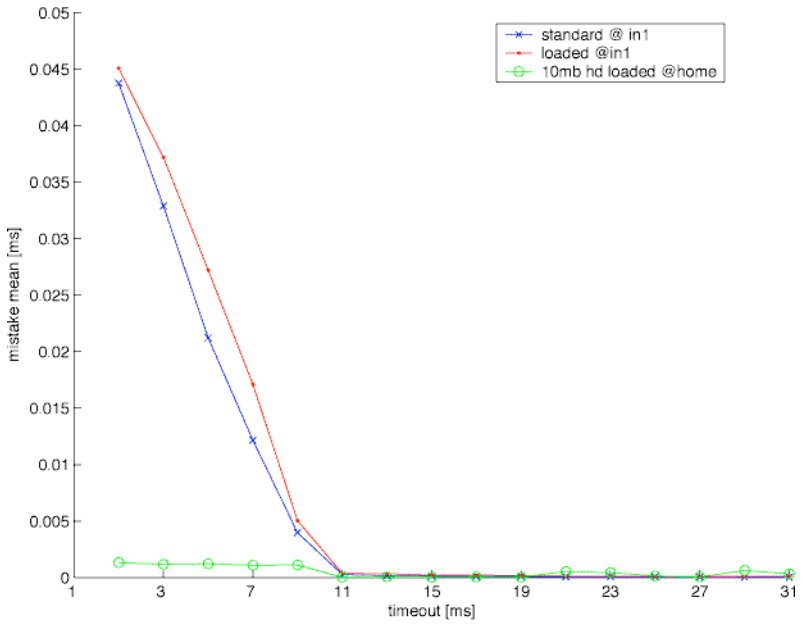

Figure 2 : Durée moyenne des suspicions erronées en fonction du *timeout*.

La Figure 2 illustre les performances du service dans un réseau local. Nous avons mesuré la durée de suspicions d'un nœud sans défaillances en faisant varier la durée des timeout – cela nous permet

de mesure la qualité de la détection de panne en fonction de l'agressivité des *timeouts*. La détection se fait uniquement au moyen de messages *heartbeat* envoyés toutes les 5 ms. Les messages de *heartbeat* sont envoyés avec des trappes SNMP, elle même encapsulées dans des messages UDP. On voit que le taux d'erreur est dramatiquement réduit lorsque le timeout correspond à l'envoi de deux *heartbeats*. Cela est dû au fait que les chances de perdre deux messages UDP consécutifs sur un réseau local sont très faibles. Le temps de détection obtenu est de l'ordre de 12 ms, ce qui permet l'implémentation d'applications très responsives.

Une implémentation complète est en cours basée sur le système SNMP4J. Elle inclut des interfaces standardisées, un système de message configurable, ainsi que des politiques de timeout dynamiques. Ce prototype sera validé en l'intégrant au système Neko [7] ainsi que dans des primitives de communication point-à-point.

Conclusion

Le service de détection SNMP-FD offre un service de détection doté d'une interface standardisée. Il permet d'offrir aux développeurs le résultat de la recherche en matière de détection de fautes sans qu'ils aient à se préoccuper des complexités des techniques ou d'une interface particulière. L'utilisation d'un système léger permet de garder de bonnes performances et l'intégration avec les équipements réseau et offre les avantages d'une technologie standard et mature.

Références

[1] M. Wiesmann, X. Défago, and A. Schiper. **Group communication based on standard interfaces.** In *Proceedings of the IEEE International Symposium on Network Computing and Applications (NCA-03)*, pages 140–147, Cambridge, MA, USA, 2003.

[2] F. C. Gärtner. **A gentle introduction to failure detectors and related problems**. Technical Report TUD-BS-2001-01, Darmstadt University of Technology, Department of Computer Science, 2001.

[3] R. de Araújo Macêdo and F. Ramon Lima e Lima. **Improving the quality of service of failure detectors with SNMP and artificial neural networks**. In RS, Brazil, 2004. *Anais do 22o. Simpósio Brasileiro de Redes de Computadores*, pages 583–586, Gramado,

[4] F. Reichenbach. **Service SNMP de détection de faute pour des systèmes répartis**. Travail de Diplôme, École Polytechnique Fédérale de Lausanne, Suisse, Février 2002.

[5] M. Müller. **Performance evaluation of a failure detector using SNMP**. Project de Semestre, École Polytechnique Fédérale de Lausanne, Suisse, Février 2004.

[6] F. Zennaro. **Mesure de performance d'un service de détection de pannes SNMP**. Project de Semestre, École Polytechnique Fédérale de Lausanne, Suisse, Février 2004.

[7] P. Urbán, X. Défago and A. Schiper. **Neko: A single environment to simulate and prototype distributed algorithms**. *Journal of Information Science and Engineering*, 18(6):981–997, November 2002.

Remerciements

Je tiens à remercier Stefan Pleisch pour m'avoir aidé durant le stade initial de ce projet ainsi que Xavier Défago, Péter Urbán, Naohiro Hayashibara pour les discussions intéressantes que nous avons eues.