NSY 107 : Intégration des systèmes client-serveur
Correction de l'examen du 01/07/2006

Partie I : QCM

1) Il est impossible de concevoir un programme client et un serveur mélangeant le mode « datagramme » et le mode « connecté » (ces deux modes étant incompatibles) ..... Vrai ❑ **Faux ☑**

Commentaire : en effet, il est tout à fait possible de créer un serveur qui :

- offre le même service en UDP et en TCP sur le même numéro de port,
- ou encore, d'inventer un protocole client/serveur qui dialogue en UDP, et qui ouvre de temps en temps des liaisons TCP pour transmettre de longs flux ; ou inversement, qui ouvrent un canal TCP, et qui échangent des informations de signalisation en UDP. Pour conclure : seule votre imagination est une limite…

2) Les « *web services* » sont des mécanismes client/serveur dans lesquels les flux échangés sont du XML ? .... Vrai ❑ **Faux ☑**

Commentaire : les *web services* s'appuient sur le protocole HTTP ; autrement dit, les données échangées sont encapsulées dans du HTTP. Mais rien n'est dit sur la nature de ce qui est échangé : ça peut être du XML, mais aussi tout autre chose (du HTML, un texte en ASCII, etc.).

3) Lors de l'évaluation de la sécurité de vos serveurs au niveau physique, les constructeurs fournissent des informations statistiques pouvant vous aider à prendre une décision ?..... **Vrai ☑** Faux ❑

Commentaire : c'est le rôle du MTBF (Mean Time Between Failure).

4) Dans un réseau Ethernet utilisant le protocole spaning tree (STP v. 1), faire une boucle entre tous les commutateurs augmente la redondance des liens, donc, le niveau de sécurité ?..... Vrai ❑ **Faux ☑**

Commentaire : c'est effectivement ce qu'on pourrait penser : en faisant une boucle, si un lien se casse, il suffit de passer par l'autre chemin de la boucle. Mais en pratique, avec STP, faire une boucle a des conséquences catastrophiques : les commutateurs n'arrivent plus à cartographier le réseau, les tables d'adresses mac « *rebondissent* » entre tous les commutateurs, et le réseau tombe en quelques instants.

5) Dans l'architecture CORBA, IDL est un langage de description des interfaces des différents objets qui, une fois compilé, génère automatiquement le squelette de certains programmes ?..... **Vrai ☑** Faux ❑

Commentaire : c'est effectivement un des atouts de CORBA. Il est possible de définir les APIs des différents objets dans ce langage IDL, et un compilateur IDL génère le squelette de la future solution, dans le langage que vous voulez (Cobol ☹, C, C++, ADA, etc.).

6) Il est possible de compiler du code écrit en JAVA dans l'environnement de programmation Microsoft .NET ? .... **Vrai ☑** Faux ❑

Commentaire : tout à fait exact. Microsoft a sorti le compilateur J# (prononcer « J chart ») à cet effet. Evidement, l'inverse est impossible (compiler du C# ou du Visual Basic dans l'environnement de développement Java).

7) Un middleware est un connecteur permettant de rendre un programme utilisant des requêtes SQL indépendant du système de gestion de base de données (SGBD) utilisé ?..... Vrai ❑ **Faux ☑**

Commentaire : c'est justement l'inverse. La phrase correcte est « *Un connecteur est un middleware permettant de rendre un programme utilisant des requêtes SQL indépendant du SGBD utilisé* ». Le connecteur est un middleware, mais tous les middleware ne sont pas des connecteurs (de façon générale, un middleware est un adaptateur de protocole entre un client et un serveur).

8) Dans une architecture N-tiers, les composantes suivantes sont indispensables *(cochez les réponses vraies)* : .... Firewall ❑ Navigateur web ❑ **SGBD ☑** Routeur ❑ **Logiciel client ☑**

Commentaire :

- le firewall n'est indiqué que si le client est sur un réseau public (Internet par exemple). Il est déjà moins indispensable dans un LAN isolé ;
- le navigateur web est un client possible. Mais il y a des architectures ou le client n'est pas un navigateur, mais un programme spécifique ;
- évidemment, SGBD et logiciel client sont deux couches du modèle 3-tiers ; alors, à fortiori, du N-tiers.

- pas besoin de routeur si tous les éléments (clients, serveurs, SGBD) sont sur le même réseau IP. D'ailleurs, de façon générale, l'architecture N-tiers n'impose rien du point de vue réseau.

9) Un logiciel antivirus assurant une protection « temps réel » est capable de bloquer toutes les attaques virales dont il possède la signature ? ..... Vrai ❑ **Faux ☑**

Commentaire : malheureusement non, pas *toutes* les attaques. Ils vérifient les accès aux fichiers, les flux réseau, etc. Mais nous avons vu par exemple les attaques par « dépassement de pile ». Les antivirus temps réel ne savent pas détecter les infiltrations de ce type. Tout au plus, ils détectent au bout d'un instant que la machine est infectée, ils arrivent parfois même à éradiquer le virus. Mais il s'agit alors d'un traitement à posteriori, pas d'une détection/blocage en temps réel.

10) L'utilisation du protocole XDR permet à des micro-ordinateurs PC ou Macintosh de dialoguer avec des serveurs RS-6000 ? ..... **Vrai ☑** Faux ❑

Commentaire : pourquoi pas ? En effet, XDR permet les échanges de données entre machines ayant des représentations internes différentes (couche présentation du modèle OSI). Alors pourquoi pas PC/RS6000, ou Macintosh/RS6000. PC, Mac, et RS6000 ne sont que 3 exemples parmi tant d'autres possibles.

Partie II : Questions ouvertes (5 points)

1) Faire un schéma illustrant le parallèle qu'il est possible de faire entre le modèle OSI et l'utilisation des concepts RPC et XDR *(1 pt)*. Dans ce schéma, vu du programmeur, quelle est la différence entre l'appel aux fonctions de la couche « XDR » et l'appel aux fonctions de toutes les autres couches inférieures *(1/2 pt)* ?

Pour le schéma, la question était simple, la réponse étant fournie dans le tout premier cours (diapo 14). Il y avait juste à faire un peu de filtrage (notez qu'il fallait bien mettre TCP et UDP en couche transport ; sinon, ½ pt) :

	Modèle OSI		Modèle Sun RPC
7	Application	⇢	Programmes Applicatifs
6	Présentation	⇢	XDR
5	Session	⇢	RPC
4	Transport	⇢	UDP | TCP
3	Réseau	⇢	IP
2	Liaison	⇢	Ethernet, ATM…
1	Physique	⇢	Électronique, lumière…

La deuxième question faisait plus appel à votre réflexion : lorsque vous programmez une application, et que vous faites appel aux fonctions de la bibliothèque XDR, ces fonctions font le transcodage des données, puis vous rendent la main. C'est alors à vous d'appeler ensuite les fonctions de la couche inférieure (RPC). Alors que tous les autres appels aux fonctions des couches 5 et inférieures font eux même appel à la couche inférieure, et ce, de couche en couche, de façon automatique (RPC appelle UDP ou TCP qui appelle IP qui utilise la carte réseau…). C'est d'ailleurs une des raisons qui fait que souvent, les couches 5, 6, et 7 sont regroupées en une seule.

2) Vous êtes nommé cadre supérieur dans une grande entreprise de production de produits manufacturés. Après une brève analyse de l'existant, vous vous rendez compte que votre société n'a pas de politique de sécurité de son système d'information. Ayant suivi le cours NSY107, vous savez qu'une telle situation est dangereuse. Aussi, vous décidez de mettre en place une politique de sécurité. Quelles seront les 4 premières étapes de ce projet *(2 pts)* ?

Etape 1 : faire adhérer cette idée par l'équipe de direction, et leur faire inscrire une ligne budgétaire (les décideurs doivent cautionner le projet, et mettre les moyens en face ; sinon, inutile d'aller plus loin).

Etape 2 : désigner le RSSI (responsable sécurité du système d'information). C'est lui qui portera le projet.

Etape 3 : évaluer les risques : faire l'inventaire exhaustif des contraintes réglementaires et législatives entourant votre activité, et voir les conséquences en terme de système d'information (traçabilité, obligations d'archivage, etc.). Faire le même travail avec votre production (en quelle mesure une panne informatique a des effets sur la production de votre usine ? Sur les relations commerciales ? Sur la santé de votre personnel, sur l'environnement, etc.).

Etape 4 : plusieurs réponses/nuances sont possibles et seront acceptées : bilan de l'existant, définir un schéma cible, définir un calendrier d'actions, faire réaliser un audit externe, etc.

3) En général, les progiciels permettent l'édition d'états prédéfinis. Et souvent, il est possible de faire des requêtes dans le système de gestion de base de données utilisée par ces progiciels, vous permettant ainsi de calculer certains indicateurs. Pourtant, une grande majorité d'entreprises met en place des « *datawarehouse* » et utilise des

langages de 4ème génération (L4G) pour faire du « *datamining* ». Pourquoi – quels sont les avantages de cette dernière solution – *(1,5 pt)* ?

Trois bons arguments permettront de faire le plein des 1,5 points. En voici 3 (il y a certainement d'autres qui seront aussi acceptés) :

- faire des requêtes - qui peuvent être complexes - directement sur un SGBD en production peut s'avérer gourmand en ressources machines, et pénaliser les utilisateurs. Alors qu'en recopiant les données sur une base annexe, il est alors possible de faire de gros traitement sur cette base sans perturber le travail des utilisateurs ;
- les progiciels ne contiennent que les données de leur domaine de compétences (le logiciel de compta ne contient que les données financières, le logiciel de GRH que les données sur le personnel, etc.). Or, les datawarehouse collectent des données depuis plusieurs applicatifs. Il devient alors possible de croiser des informations de domaines fonctionnels différents ;
- les L4G proposent des outils de présentation, de calcul, etc. Il est donc inutile de réinventer la roue à chaque fois que vous souhaitez construire un tableau de bord.

Partie III : Problème technique (5 points)

Des investisseurs souhaitent créer une société de vente par correspondance de billets d'avions « *low cost* » sur Internet : « *lowcostfly.com* ». Au début de l'histoire de cette société, la montée en charge de l'activité risque d'être chaotique ; en terme informatique, le site aura à s'adapter à des augmentations soudaines du nombre de visites et de transactions.

Vous êtes désigné pour proposer une architecture informatique de ce nouveau site. Dans votre cahier des charges, les contraintes qui vous sont imposées sont :

- pour anticiper ces futures crises de croissance, et pour rendre les différents développements le plus indépendant possible de solutions commerciales particulières, vous devez vous orienter vers une architecture N-tiers ;
- pour connaître les promotions et offres de dernière minute faites par les compagnies aériennes, votre système devra interroger très régulièrement leurs catalogues en ligne (sachant qu'il n'existe pas un protocole unique pour interroger ces catalogues, chaque entreprise ayant sa propre solution informatique, incompatible avec celle du concurrent). Les commandes enregistrées sur votre site devront immédiatement être transférées auprès de la compagnie concernée ;
- enfin, votre commerce étant très lié à la disponibilité de votre site, les investisseurs sont près à supporter les coûts liés à la duplication de la salle machine sur deux sites éloignés.

Lors de l'étude préliminaire, on vous demande de proposer un schéma de l'architecture cible répondant à ces contraintes et objectifs. N'hésitez pas à compléter ce schéma de quelques mots pour justifier les choix qui ne semblent pas nécessairement évidents, ou pour expliquer le rôle de certains éléments.

Remarque : à ce stade du projet, on vous demande une architecture générale ; on ne vous demande pas encore de faire de choix de langage de développement, de modèle de programmation réseau, etc. Vous devez simplement lister et positionner les grands « services » que vous aurez à mettre en place.

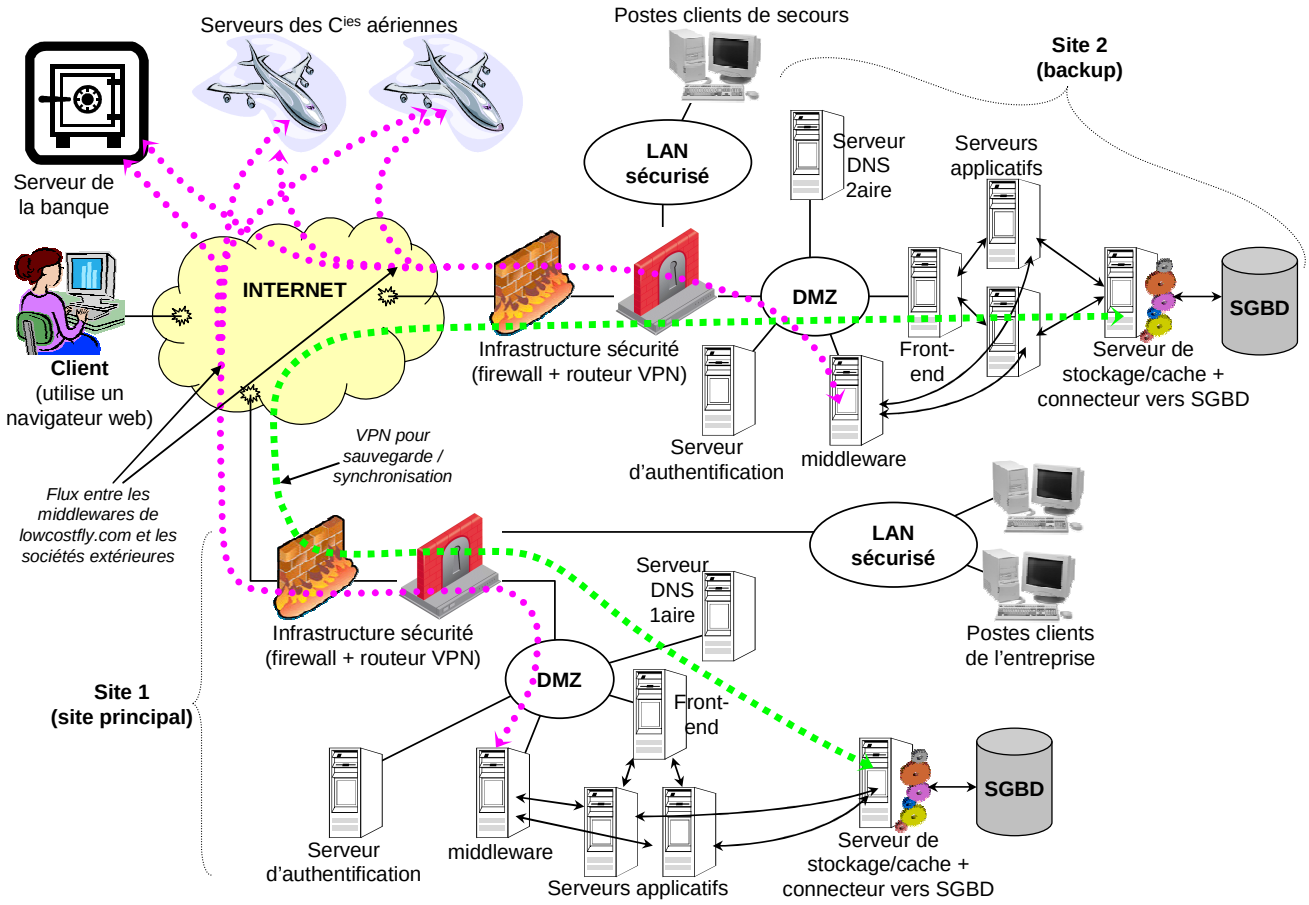
Voici (ci-dessous) un schéma possible. Evidemment, il existe quelques variantes à ce schéma, qui seront elles aussi acceptées. Voici quelques explications :

- tout d'abord, il faut enregistrer le nom de domaine « lowcostfly.com » auprès d'un « *registar* ». Aussi, il faudra dans votre réseau final un serveur DNS principal, et par sécurité, un serveur DNS secondaire ;
- pour gérer les autorisations d'accès aux ressources partagées et aux différents applicatifs, la mise en place d'un serveur d'authentification (un sur chaque site) est appropriée.
- aujourd'hui, il serait très risqué de monter un site web commercial sans mettre l'infrastructure de sécurité ad hoc (routeur VPN, firewall, etc). Le firewall protègera le réseau interne des attaques pouvant provenir d'Internet. Le routeur VPN permet de créer un réseau privé virtuel (lien totalement sécurisé) entre les deux sites permettra :
    - la synchronisation des DNS,
    - la synchronisation des serveurs d'authentification,
    - la synchronisation des SGBD,
    - les sauvegardes des fichiers d'un site sur un autre,
    - etc.
- le cœur de l'architecture N-tiers :

- le client, qui utilise un navigateur web ;
- les serveurs applicatifs, la charge étant répartie par un « front-end » (remarque : il est possible de détailler, en détaillant cette grappe de serveurs applicatifs en deux classes : les applications métier – i.e. la gestion du site lowcostfly.com –, et les services « standards » ou « communs » – comme le serveur d'emails, le serveur d'impression, etc. –) ;
- le serveur de stockage, qui optimise les accès aux objets stockés à l'aide d'un cache, et qui accède au SGBD à travers un connecteur (afin de rendre les applicatifs les plus indépendants du SGBD mis en place).

- Un middleware (sur chaque site), dont le rôle sera :
  - De télécharger régulièrement les tarifs des compagnies aériennes depuis leurs systèmes d'information, et d'y enregistrer les commandes (notion de B2B : business to business),
  - De permettre le télépaiement avec un système bancaire.

Le résultat final doit ressembler à :



Partie IV : Note de synthèse (5 points)

A la lecture de ces trois extrais d'article, et selon vos connaissances, indiquez les points communs/différences, avantages/inconvénients des 2 générations de RPC : XML-RPC et SOAP, versus les RPC de Sun couplé à XDR.

SOAP, XML-RPC & REST: différences et intérêts - *05/11/2004*

[…] **XML-RPC**
Le protocole XML-RPC peut donc être pris comme l'ancêtre de SOAP : il a en tous cas évolué d'une première version de SOAP, et modifié pour correspondre à la vision de Dave Winer. [...]

SOAP
Créé sous les hospices de Microsoft et toujours soutenu par ce dernier, le protocole SOAP se montre bien plus populaire et utilisé que son camarade XML-RPC - principalement grâce au soutient du W3C, mainteneur de la spécification depuis la version 1.2 du protocole, publiée en 2003. De fait, aujourd'hui, SOAP est un standard de facto du monde des services Web […]

Conclusion
[…] Mais aujourd'hui, tous les langages modernes disposent de fonctionnalités leur permettant d'exploiter des services Web type SOAP ou XML-RPC sans se soucier de la verbosité de leurs messages sortants et entrants. C'est d'ailleurs la raison pour laquelle ces deux techniques ont tant de succès vis-à-vis de REST : les outils sont disponibles, tandis que REST nécessite de mettre en place ses propres méthodes (même si le protocole, lui existe déjà).
De SOAP ou XML-RPC, le choix des développeurs tend à se porter sur la première méthode, du fait de certains défauts dans la spécification de la seconde : manque de précisions, confusions sur certains aspects (support Unicode, notamment), mots de passe transmis en clair… […]

Xavier Borderie, JDN Développeurs.

Portmap et les RPC (Remote Procedure Call)
05/01/2005, D. David (Coredump)

1. Les RPC
Les RPC […] permettent l'exécution de procédures sur une machine distante. […] Les procédures (ou fonctions) RPC sont regroupées en programmes et identifiées par des numéros. Les programmes se voient aussi attribuer un numéro ainsi qu'un numéro de version. C'est par le biais de ce triplet qu'un client peut appeler une procédure particulière. Les numéros sont attribués d'une façon stricte (idem ports TCP et UDP). Ainsi mountd se voit attribuer le numéro 100005.

2. XDR
Conjointement aux RPC et au-dessus (dans le modèle OSI), on utilise le protocole XDR (eXternal Data Representation). XDR gère la mise en forme des données[…]. Il définit un standard de représentation des types sur le réseau, afin notamment de palier la multiplicité des représentations utilisées (big endien, little endien, ...).

3. Portmap
[…] Ces appels sont gérés par un standard (le processus portmap) via les numéros vus plus haut. Ces numéros sont recensés dans le fichier /etc/rpc. Ce fichier contient les services constructeurs, ce n'est pas un fichier de configuration, il joue le même rôle que /etc/services dans la correspondance nom d'application/port associé pour la programmation RPC. […]

XML, Soap, WSDL et UDDI : les composants des services web - 01 Réseaux, le 01/12/2002

[…]En pratique, sur un réseau, un analyseur de trames IP verra la chose suivante : HTTP, qui encapsule Soap, qui encapsule les flux métiers. Mais XML, et Soap n'ont pas de sécurité intégrée. Il faut donc les compléter. La première étape est SAML (Security assertion markup language) , qui fixe un cadre XML pour l'échange des informations de sécurité.[…]

La note de synthèse est un exercice de style où souvent, la forme a presque autant d'importance que le fond. Une des difficultés est de savoir s'il faut se contenter des informations contenues dans les documents (c'est plutôt le cas pour les examens « littéraires »), ou s'il est autorisé d'utiliser sa culture personnelle (souvent le cas pour les examens plus techniques). Quand aucune règle du jeu n'est donnée, force est de constater que la règle est souvent « notateur dépendant ». Heureusement, ici, la règle du jeu était donnée : « […]et selon vos connaissances[…] ». Pour autant, le sujet étant assez fermé, et les textes donnés pointant sur la quasi-totalité des arguments, il n'y a pas beaucoup à aller chercher dans « nos connaissances ».

Pour la forme, la note de synthèse se rédige comme des rédactions ou dissertations classiques : thèse/antithèse/synthèse, ou encore, faire dialoguer les textes entre eux, etc. Une chose est importante : votre texte doit toujours avoir un plan évident. A savoir :

- une introduction, qui rappelle la problématique et éventuellement, pose quelques définitions. Le plan de ce qui suit doit être annoncé à l'issue de l'introduction. C'est d'ailleurs une des raisons pour laquelle l'introduction doit souvent être rédigée en dernier ;
- un développement (argumenté),
- et une conclusion.

Revenons à notre sujet. On nous demande de comparer deux générations de technologies : Sun RPC, et le couple XML-RPC et SOAP. Normalement, les textes doivent pointer sur les sujets que nous avons à reprendre. Lisons-les :

1) le texte « SOAP, XML-RPC, et REST : … » (que nous noterons par la suite [Texte*1*]) : SOAP et XML-RPC ont des origines communes. Il s'agit de service web (autrement dit, il s'agit de web services ; nous savons aussi que ces web services échangent du XML ; le texte précise d'ailleurs qu'il s'agit de protocoles « verbeux » - comprendre

bavards -). On apprend que XML-RPC laisse transiter les mots de passe en clair, contrairement à SOAP. Sous-entendu : ce qui n'est pas un mot de passe, avec ces deux technologies, transite en clair. C'est plutôt moyen en termes de sécurité.

2) le texte « Portmap et les RPC… » (que nous noterons par la suite [Texte*2*]) : RPC utilise un mécanisme qui permet de rendre disponible la liste de services proposés, au travers d'un serveur : le « portmap » (qui n'existe pas avec SOUP et XML-RPC). Les échanges entre client et serveur se fait via la couche de présentation des données : XDR (qui n'est pas du XML, et qui n'est pas chiffré non plus).

3) le texte « XML, Soap, WSDL, et UDDI : … » (que nous noterons par la suite [Texte*3*]) : on a confirmation que XML-RPC et SOAP sont des web services (ils s'appuient sur le protocole HTTP). On a confirmation que ces mécanismes ne sont pas sécurisés : pas de chiffrement, ni de mécanisme d'authentification (on nous dit même qu'il faut ajouter une couche SAML pour sécuriser tout ça). Remarque : les plus cultivés d'entre vous savent aussi (information donnée en cours) que les Sun RPC sont de moins en moins activés sur les différents serveurs, ce mécanisme ouvrant souvent de nombreuses failles de sécurité.

Ce travail de lecture terminé, il ne reste plus qu'à établir un plan détaillé (ce que nous allons faire ci-dessous), et à transformer ce plan en joli français (ce qui n'est pas fait dans la présente correction ; les plus courageux des lecteurs peuvent s'y coller).

Plan détaillé (d'une correction possible) :

Introduction : les RPC de Sun, ainsi que XML-RPC et SOAP sont des mécanismes d'appel de procédure à distance. Nous allons voir que les premiers, de conception ancienne, avaient pourtant quelques avantages, qui ne se retrouvent plus dans les seconds. Nous verrons ensuite que les philosophies de ces deux générations de RPC sont assez différentes, ce qui engendre des conséquences en terme de bande passante et de performance. Enfin, nous verrons que tous ces RPC, peut-être pour des raisons parfois différentes, sont peu sécurisés.

Partie 1 : Dans [Texte *2*], il est rappelé que les Sun-RPC doivent s'enregistrer auprès d'un service : le portmapper. Il est ainsi possible, en interrogeant ce portmapper, de savoir quels sont les services proposés par un serveur. Ce mécanisme n'existe plus avec XML-RPC et SOAP.

Partie 2 : Lors de l'utilisation de Sun-RPC, le client et le serveur dialoguent directement, en échangeant des informations sur des sockets. Or, [Texte *3*] nous rappelle que pour XML-RPC et SOAP, le dialogue client-serveur se fait en utilisant le protocole HTTP (les requêtes sont encapsulées dans des requêtes HTTP). Or, nous savons que ces techniques d'encapsulation sont moins performantes. De plus, les Sun-RPC utilisent XDR comme format de représentation des données. Ce format est bien plus condensé, alors que XML est plus bavard, comme rappelé dans [Texte *1*]. Ce même texte nous indique aussi que cet inconvénient est contrebalancé par le fait que les langages modernes intègrent de base les mécanismes permettant de réaliser les appels à ces RPC. De plus, avec son mécanisme de balises, on sait que XML est plus structuré (il permet une vue en arborescence des données).

Partie 3 : [Texte *3*] nous rappelle que XML-RPC et SOAP sont des mécanismes qui, intrinsèquement, se sont pas sécurisés (pas de chiffrement, ni d'authentification). Ce même texte ajoute qu'il faut intégrer une couche supplémentaire (SAML) pour sécuriser les échanges. Pire : [Texte 1] nous rappelle même que les mots de passe sont transmis en clair avec XML-RPC. Pour autant, nous savons aussi que Sun-RPC est décrié, à cause des problèmes de sécurité induits à sa mise en place.

Conclusion : Les trois technologies permettent d'implémenter des RPC. Sun-RPC, bien qu'ancien, proposait quelques avantages : accessibilité de la liste des services proposés grâce au portmapper, et protocoles peu bavard, plus efficace. Pour autant, parce que les mécanismes implémentés dans les langages modernes simplifient le travail du programmeur, les RPC basés sur les webservices et échangeant du XML (plus structuré qu'un flux XDR) sont devenus populaires. Il faut néanmoins garder à l'esprit que toutes ces techniques ne doivent être utilisées que pour des applications peu critiques, en raison de leur manque de sécurité.