Application du protocole SVC  
dans la communication d’un réseau  
de transport public

Préface

Contents

[I. Problématique 4](#_Toc460267659)

[II. Analyse 5](#_Toc460267660)

[1. La nature de la communication entre les bus et le serveur 5](#_Toc460267661)

[2. Les exigences d’un protocole de communication spécifique 6](#_Toc460267662)

[3. Les solutions existants et la motivation d’un nouveau protocole 7](#_Toc460267663)

[III. Solution 8](#_Toc460267664)

[1. L’architecture de la solution 8](#_Toc460267665)

[2. Les contraintes sécurisées 8](#_Toc460267666)

# Problématique

La Régie des Transports Marseillais (RTM) est la société qui est en charge du service de transportation à Marseille – une de trois plus grandes villes de la France. Dans les efforts d’évolution et d’amélioration de ses services, en particulier le réseau de transport en bus, la RTM a besoin d’y mettre en place un système de surveillance, qui est une des exigences de la spécification EBSF.

Pour supporter la communication entre les bus et les serveurs, des réseaux Internet hétérogènes seront utilisés (wifi, 3G, 4G), sur lesquels s’appliquent un protocole de communication sécurisé. Tels protocoles existent sur le marché, mais rien n’a été conçu pour ce problème spécifique.

Notre mission est donc d’analyser la nature de la communication, les avantages et désavantages des solutions existant et de donner la solution la plus adaptée à ce besoin.

# Analyse

## La nature de la communication entre les bus et le serveur

La communication dans le réseau de transport est un type de communication spécifique. Elle subit à des contraintes et obligations faites de la nature du service.

* **Basculement rapide** **de connexion** : en raison du mouvement des bus, la connexion vers le serveur est souvent interrompue quand le bus sort d’une zone de couverture d’une antenne vers celle d’une autre, ou empêchée par de grands obstacles physiques. Le changement du type de réseaux peut aussi se produire au cas où le support du réseau courant ne peut pas desservir toutes les stations qui sont au tour d’un point d’accès.
* **Gros volume de données** : les données cueillies pendant les trajets du bus, concernant le statut de tous les équipements, la position cartographique du véhicule et les points d’accès, et notamment le vidéo-surveillance, sont montées et envoyées vers le serveur. Non seulement un réseau de haut débit mais aussi un bon protocole de transmission qui peut réduire la surcharge de bande passante est requis.
* **Diversité de types et de priorités de données :** les données montées viennent de sources différentes, certaines sont capables de tolérer des erreurs (par exemple quelques trames de vidéo-surveillance), autres requièrent strictement être livrées. On parle aussi des priorités d’envoi, sur lesquelles les données sont traitées et envoyées dans le cas de congestion.
* **Temps réel :** toutes les données de surveillance doivent être transmises au serveur dès qu’elles sont disponibles, pour que les contrôleurs aient des réactions à temps en cas d’incidence.
* **Sécurisée :** les informations collectées dans le bus, y compris les images des passagers sont liées à la vie privée, doivent être transmises et exploitées de façon sécurisée. Une méthode de chiffrement est indispensable dans l’ensemble de solution proposée.

## Les exigences d’un protocole de communication spécifique

## Les solutions existants et la motivation d’un nouveau protocole

Les protocoles sécurisés qu’on se voit de nos jours sont basées essentiellement au TCP : TLS, SSH, OpenVPN, etc…

Déjà, le TCP n’est pas un choix idéal pour notre solution. Il ne satisfait pas la première caractéristique de la connexion, tant que l’établissement et la reprise de la connexion TCP prend toujours du temps considérable. De plus, la garantie de livraison de paquet n’est pas nécessaire en permanence, car dans plupart de temps, les données sont tolérantes aux pertes. La compromise entre la fiabilité et la rapidité du TCP augmente la latence de la connexion, le rend moins efficace de l’utiliser dans un contexte temps réel.

On peut par contre trouver quelques versions UDP de ces protocoles ci-dessus (comme DTLS), ou ces protocoles offrent une option UDP eux-mêmes. Pour garder la même opérabilité qu’en TCP, des modifications supplémentaires sont introduites. Dans l’optique de minimiser la complexité, ce n’est pas non plus une bonne décision.

D’ailleurs, comme indiqué dans la recherche de G. Risterucci et al. [[1]](#footnote-1), les protocoles mentionnés s’imposent encore des limites contre la protection d’identité de client ou l’exposition des informations sensibles.

Dans cette vision née une idée de SVC, un nouveau protocole sécurisé qui surmonte ces limites en proposant une négociation simple et un découplage entre l’authentification et le chiffrement. SVC implémente les algorithmes de chiffrement les plus récents sans utiliser des sources externes, cela facilitera le processus d’administration et maintenance.

Et pourtant, la version origine du SVC a besoin de quelques améliorations pour mieux s’adapter au problème posé. Dans la section suivante, on propose une version légèrement modifiée du SVC avec la conception de l’ensemble de la solution.

## Les désavantages de la version original du SVC

On a trouvé beaucoup d’idées innovantes dans la conception du SVC. Parmi lesquelles, on a choisi à développer l’idée de découplage l’authentification et l’autorisation, et l’idée de protection d’identité du client. Et pourtant, SVC présente dans la conception des points faibles à améliorer.

Dans la version originale proposée par ses auteurs, le protocole commence par une requête depuis le serveur vers le client (après un succès de connexion). Cette approche est justifiée comme une façon de réduire la duration de la négociation. Nous, on n’est pas d’accord avec cette explication. À compter la phase d’initialisation de connexion, SVC consommes 4 exchanges de message.

Le problème le plus grave c’est le fait d’avoir l’étape d’échange de clé commence par le serveur, rend le protocole sensible aux attaques par rejeu. La seule façon pour l’éviter, c’est de rajouter du timestamps, qui ne sera pas une bonne idée.

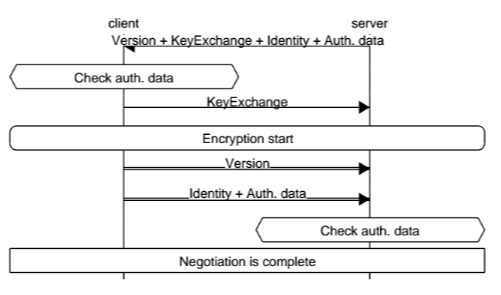


Figure 3.1 – Echange de messages dans la version originale du SVC

# Solution

## La version modifiée du SVC

Avec seulement 3 échanges, on propose une nouvelle approche basé sur le principe du « serrement en 3 phases » qui garde encore les idées basiques du SVC.

Dans le diagramme suivant, on distingue les données « dans les crochets », avec celles qui sont en dehors. Les dernières sont des données de la négociation DH-STS, qui assurent l’autorisation du service, deviennent optionnelles en raison du découplage de service (à expliquer dans la section suivante). Les premières sont des données d’authentification, dépendantes de l’application qui utilise SVC.

Dans la vérification de l’authenticité, l’approche « challenge – proof » est introduite pour éviter tout type d’attaque par rejeu. Le DH-STS est forcément imperméable, seulement vulnérable à quelques attaques du type « unknown key-share »[[2]](#footnote-2).

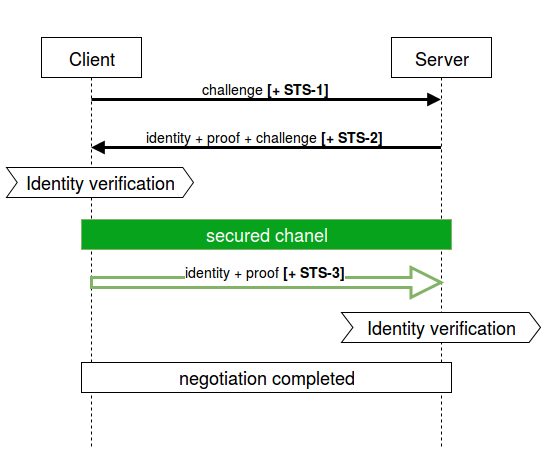


Figure 3.2 – Echange de messages dans la version modifiée du SVC

À comparer avec la version originale, la nouvelle montre des avantages supplémentaires:

* Moins de nombre des échanges (3 contre 4)
* Résistant à l’attaque par rejeu
* Découplage de service

## L’architecture service-applications

À la naissance, SVC est vise à servir une communication interne, c’est-à-dire une communication entre les instances d’une même application. Dans l’environnement multitâche, cette conception montre des défauts:

* chaque connexion doit manager ses propres (mais avec une même politique) paramètres de sécurité, chiffrement. Si on a plusieurs connexions vers une même hôte, toutes les étapes de négociation se refont.
* La mise à jour des protocoles de chiffrement du SVC impose la recompilation de toutes les applications qui l’utilisent.

L’idée est de découper la couche « application », qui est en charge de l’authentification et la communication de données, avec une seule instance de la couche « service », qui s’occupe du chiffrement et la négociation. Comme ça, on remédie tout de suite les problèmes posés :

* Le service manage toutes les connexions depuis et vers des hôtes. Quand une nouvelle connexion vers une même hôte est détectée, on profite d’un canal déjà sécurisé pour la négociation, laisse passer les échanges de clés.
* La mise à jour se fait en simplement replaçant et redémarrant l’instance du service. Toutes les applications restent intouchées.

Dans le diagramme ci-dessous se trouvera l’architecture de la solution :

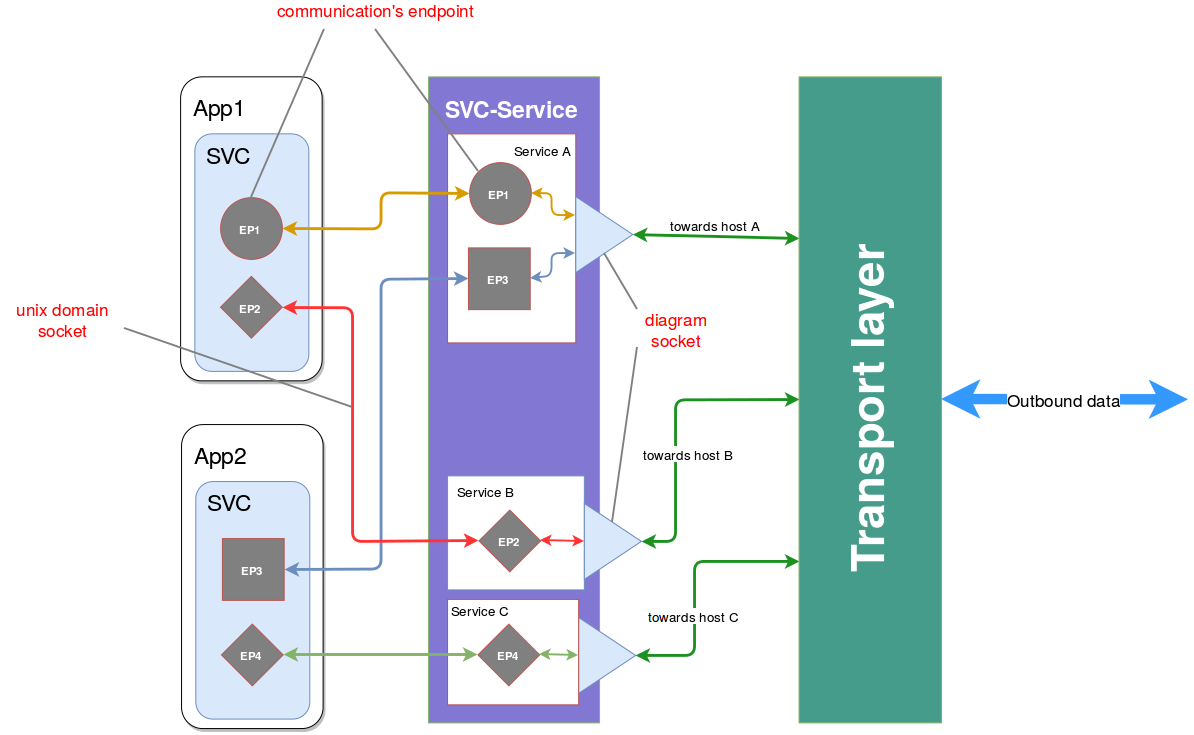


Figure 3.3 – L’architecture service-application du SVC

## Les contraintes sécurisées

1. G. Risterucci, T. Muntean, L. Mugwaneza. *A new Secure Virtual Connector approach for communication within large distributed systems*. ERISCS Research Group. [↑](#footnote-ref-1)
2. Blake-Wilson, S.; Menezes, A. (1999), "Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol", *Public Key Cryptography*, Lecture Notes in Computer Science, **1560**, Springer, pp. 154–170 [↑](#footnote-ref-2)