A heuristic algorithm imitating social behaviours for intelligent routing of application flows

Jean-Philippe Fauvelle

A heuristic algorithm imitating social behaviours for intelligent routing of application flows

Jean-Philippe Fauvelle
Contact: https://www.linkedin.com/in/jpfauvelle/

Abstract: The intelligent routing of application flows between, on the one hand, a fleet consisting of many equipments (fixed or mobile) equipped with a heavy client-side software or a light browser providing a service to end-users, and, on the other hand, a park of servers equipped with a server-side software contributing to this service, is a complex problem. A fleet may include, for example, users' computers accessing an e-commerce site, smartphones, payment terminals, ATMs and banking services, internet boxes, shopping terminals tickets, kiosks access public service, etc. The complexity of routing is inherent in the many constraints that this function should address. First, it must be adaptable to any client-side and server-side software via a single function point that interacts at the session and kinematic level. It must also take into account the compatibility matrix of the multiple deployed versions of the software. Second, it must be adaptable to any platform via a setting of topological rules incorporating the specifics of the software in the existing architecture, and allow a logical multi-criteria segmentation of the fleet and the park via a setting of rules of affinity and exclusion between any groups of equipments and servers (domains, markets, versions, service levels, etc.). Third, it must support an unlimited fleet with horizontal scaling capacity, and must dynamically enslave the load and load transfer with consideration of the overconsumption of resources generated by the software during rerouting, in order especially to avoid the cascaded spread of massive incidents (domino effect). Fourth, when necessary, it must be able to sacrifice certain flows, temporarily, to preserve a greater number of flows. Finally, the routing must offer a real-time vision of the state of the fleet and the park, and allow the automatic enrollment in the fleet of new discovered equipments. Because existing ADC (Application Delivery Controller) solutions insufficiently meet the requirements, this paper proposes a heuristic algorithm integrated into a reverse proxy, responding to the needs. This heuristic is presented in the context of automated banking machines, but can easily be transposed to any other context. Because the complexity of the problem does not allow to calculate routes in a reasonable time, the proposed heuristic does not seek the best mathematical solution but an effective and reactive routing that imitates social behaviour and maximizes availability, usability, quality of service. This inventive heuristic was implemented and tested successfully, jointly with a second invention playing the role of a distributed database (“*A distributed, probabilistic, hysteretic, retroactive algorithm to locally weaken the CAP/Brewer’s theorem when reading data on a distributed computer system*” ⟨hal-01966396⟩ https://hal.archives-ouvertes.fr/hal-01966396). Full understanding of this paper requires expertise in computer science, software engineering and algorithmics.

Keywords: Application delivery controller (ADC), Application flow routing, Software defined networking (SDN).

SOMMAIRE

LISTE DES FIGURES

LISTE DES TABLEAUX

GLOSSAIRE

Terme	Synonymes	Signification
DAB		Automate de type Distributeur Automatique de Billets.
Enrôlement d'un équipement		L'enrôlement est le procédé par lequel un équipement qui est routé par la fonction de routage (F_ROUTE) devient visible pour la fonction de compilation des routes (F_COMPIL) : • Un équipement est dit **enrôlé** ou **enrôlé complètement** (auprès de F_COMPIL) s'il existe une route courante associée à cet équipement et si cette route définit la version de l'équipement, c'est à dire la version de son progiciel ; • L'enrôlement est dit **incomplet** ou **partiel** lorsque la route courante existe sans que ladite version y soit définie.
Équipement	Automate / ATM Client (informatique) Poste client	Équipement informatique, équipé d'un « progiciel client », à partir duquel un utilisateur consomme un service. Par exemple, un automate DAB / GAB est un équipement.
Flotte	Flotte d'équipements	Ensemble des équipements constituant la flotte.
Fonction de compilation	F_COMPIL	Fonction qui implémente l'intelligence de routage en compilant les ordres de déroutages et en les communiquant à la fonction F_ROUTE qui les exécute.
Fonction de routage	F_ROUTE	Fonction qui implémente le routage des flux entre un équipement et un serveur, en exécutant les ordres de déroutages communiqués par la fonction F_COMPIL.
GAB		Automate de type Guichet Automatique de Banque, qui permet des opérations sur comptes en plus des retraits d'argent.
IMDBMS		*In-memory database management system*. Base de données en mémoire.
KVS		*Key Value Store.* Système de stockage de données sous forme de paires (clé, valeur).
Parc	Parc de serveurs	Ensemble des serveurs en production ayant pour fonction de traiter les requêtes provenant de la flotte d'équipements.
Progiciel client	Progiciel automate	Progiciel installé sur chaque équipement.
Progiciel serveur		Progiciel installé sur chaque serveur.
RAM-Cloud		Système de données distribuées dans un nuage en mémoire.
Serveur	Serveur de traitement Serveur informatique	Serveur informatique, équipé d'un « progiciel serveur », traitant les requêtes informatiques provenant des équipements.
Utilisateur	Individu Porteur de carte	Une personne qui utilise un équipement par le biais du progiciel client installé sur ledit équipement.

Tableau 1 — Glossaire.

1 EXPOSÉ DU PROBLÈME

1.1 REMARQUES PRÉLIMINAIRES

- La problématique exposée et la solution proposée sont centrées sur un contexte d'automates bancaires et de flux HTTP/S, mais demeurent néanmoins transposables dans d'autres contextes.
- Un flux ou une étape est représenté par une pastille portant le même numéro sur toutes les figures.

1.2 RÉSUMÉ DE LA PROBLÉMATIQUE

La problématique est généralisable à toute flotte d'équipements informatiques clients, fixes ou mobiles, différenciés pas des identifiants, communiquant avec un parc de serveurs de traitement via un protocole quelconque.

Toutefois, par souci de clarté, le présent document se focalise sur un écosystème monétique informatique constitué d'une flotte d'automates bancaires (ATM) utilisés par des individus porteurs de cartes bancaires et d'un parc de serveurs traitant les requêtes informatiques au format HTTP ou HTTPS produites par lesdits automates lors de leur utilisation par lesdits individus.

Les automates sont utilisés par des individus porteurs de cartes bancaires qui effectuent des retraits d'argent (automates de type DAB ou GAB) ou des opérations sur comptes (automates de type GAB seulement).

Chaque automate est équipé d'un progiciel client qui gère la session d'un utilisateur et la cinématique de son parcours dans les différents écrans affichés pour les opérations réalisées.

Le parcours d'un utilisateur s'inscrit dans une session comportant une fin explicite (requête particulière) ou implicite (*timeout*). Afin de s'adapter aux éventuelles contraintes du progiciel, toutes les requêtes effectuées au cours d'une même session doivent, sauf incident, pouvoir être traitées par un même serveur.

Le progiciel client possède une version, compatible avec certaines versions du progiciel serveur. Plusieurs versions du progiciel client cohabitent au sein de la flotte d'automates. La référence de version du progiciel client est transmise dans les requêtes mais selon les capacités du progiciel cette transmission peut ne pas être systématique.

Chaque automate génère des requêtes dans le protocole HTTP ou HTTPS qui sont traitées par un parc de serveurs équipés d'un progiciel serveur. Ce progiciel serveur possède une version, compatible avec certaines versions du progiciel client. Plusieurs versions du progiciel serveur cohabitent au sein du parc de serveurs.

Chaque automate possède d'une part un identifiant technique nommé adresse IP qui peut changer dans le temps mais reste invariant au cours d'une session, et d'autre part un identifiant métier unique invariant. Cet identifiant métier est transmis dans les requêtes mais selon les capacités du progiciel cette transmission peut ne pas être systématique.

Le parc des serveurs possède une topologie quelconque qui peut générer des contraintes liées notamment aux capacités du progiciel serveur. Par exemple, l'existence, d'une part d'une synchronisation entre deux serveurs des données du progiciel liées au contexte de la session utilisateur, et d'autre part du délai de ladite synchronisation, conditionnent la capacité à dérouter les requêtes d'un automate entre lesdits serveurs ainsi que l'existence ou non d'une indisponibilité temporaire de l'automate suite à ce déroutage.

Chaque serveur possède une capacité de traitement limitée, associée à un nombre maximum d'automates. Cette capacité peut varier entre des serveurs hétérogènes.

Par ailleurs, lorsque les requêtes émises par un automate, initialement traitées par un premier serveur, sont déroutées vers un second serveur, le traitement de ces primo-requêtes arrivant sur le second serveur peut

consommer d'avantage de ressources, selon les capacités du progiciel. Par exemple, des déroutages nombreux simultanés pourraient occasionner une surconsommation de ressources saturant le parc de serveurs alors même que la flotte de clients ne varie pas en taille.

Le déroutage des automates doit également prendre compte le caractère d'urgence du motif de déroutage selon que l'indisponibilité d'un serveur est subie ou volontaire, ainsi que la segmentation logique de la flotte d'automates et du parc de serveurs selon des critères multiples (domaine production / pré-production, canal, partenaire, catégorie, versions, géolocalisation / région, etc.).

Enfin, la flotte d'automates doit pouvoir croître, avec une capacité de mise à l'échelle horizontale.

La gestion de cet écosystème nécessite d'effectuer un routage intelligent des flux entre la flotte d'automates et le parc de serveurs, en tenant compte de l'ensemble des contraintes et limites évoquées, tout en optimisant d'une part le taux de disponibilité de la flotte, d'autre part la continuité de service et enfin l'utilisation des ressources informatiques du parc de serveurs.

1.3 ILLUSTRATION DES FONCTIONNALITÉS ATTENDUES

1.3.1 FLOTTE, PARC, PROGICIEL

Le routage des flux applicatifs est une fonction abondement utilisée mais associée à une cohorte de problématiques que les solutions existantes de type ADP (Application Delivery Controller) adressent en partie seulement.

Afin d'illustrer ces principales difficultés et la solution de routage idéale y répondant, considérons, d'une part, une flotte d'équipements fixes ou mobiles, hébergeant un progiciel client ou un navigateur, rendant un service à des utilisateurs.

À titre d'exemple, un équipement peut être un ordinateur depuis lequel un utilisateur consulte un site de e-commerce, ou un terminal de paiement, un automate bancaire DAB/GAB, un téléphone portable, une box internet, une borne d'achat de ticket de transport ou de cinéma, un kiosque d'accès à un service public, etc. Les possibilités sont nombreuses.

Considérons, d'autre part, un parc de serveurs informatiques équipés d'un progiciel serveur compatible avec le progiciel client et contribuant au service rendu.

La flotte d'équipements peut être gigantesque, et le parc de serveurs est vraisemblablement hétérogène, distribué sur plusieurs sites, avec ou sans réplication de données entre les sites.

Quant au progiciel (côté client et côté serveur), il repose probablement sur des sessions voire des transactions qu'il convient de ne pas interrompre avant leur terme sous peine de porter atteinte au service et préjudice à l'utilisateur.

1.3.2 STRATIFICATION DES FONCTIONNALITÉS

Une solution de routage idéale peut être avantageusement stratifiée en trois couches (Figure 1) :

- Une couche « Front » dévolue au routage réflexe, fonctionnant en temps réel, résiliente à l'arrêt des deux autres couches, et disposant de son propre moteur de données ;
- Une couche « Middle » implémentant l'intelligence du routage, prenant les décisions de routage (déroutage, maintien de route, refus de route, etc.) ;
- Une couche « Back » hébergeant les fonctions de consultation et d'administration de la solution.

La solution de routage étant généralement critique, n'admet aucune perte de données (PDMA[1] = 0) pour l'ensemble de la solution et aucune indisponibilité (DMIA[2] = 0) de la fonction de routage réflexe au sein de la couche Front.

Ces exigences impliquent que la fonction de routage réflexe s'appuie, par exemple, sur un moteur de données hautement performant et résilient[3].

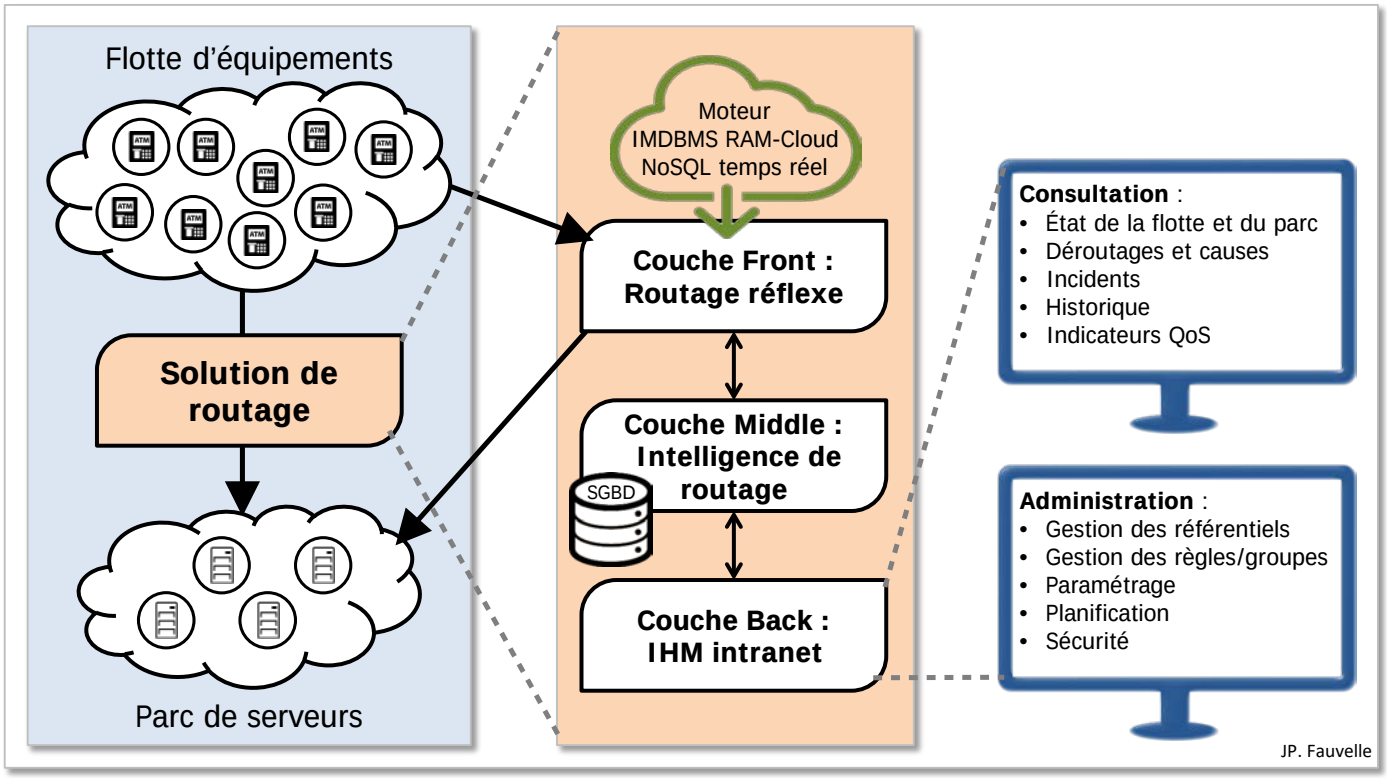

Figure 1 — Illustration des attendus / Stratification des fonctionnalités.

[1] PDMA : Perte de données maximale admissible.

[2] DMIA : Durée maximale d'indisponibilité admissible.

[3] Jean-Philippe Fauvelle. A distributed, probabilistic, hysteretic, retroactive algorithm to locally weaken the CAP/Brewer's theorem when reading data on a distributed computer system. 2020. ⟨hal-01966396⟩. https://hal.archives-ouvertes.fr/hal-01966396.

1.3.3 INTÉGRATION SANS COUTURES

La flotte et le parc s'appuient sur des infrastructures coûteuses et complexes à mettre en place et à exploiter. Afin de ne pas augmenter cette complexité et ce coût, une solution idéale doit s'intégrer dans l'existant sans nécessiter d'évolution en particulier au niveau de la flotte, laquelle peut comprendre des milliers voire des millions d'équipements.

À cette fin (Figure 2) :

- La solution est invisible pour les équipements et les serveurs, elle s'intègre via un reverse-proxy positionné entre la flotte et le parc ;
- La solution idéale de routage est agnostique vis-à-vis du progiciel, elle s'adapte à toute cinématique transactionnelle de l'utilisateur par simple analyse d'URL, elle génère une affinité de session qui demeure neutre pour le progiciel client (exemple : pas de cookie si ce progiciel client est un navigateur internet).

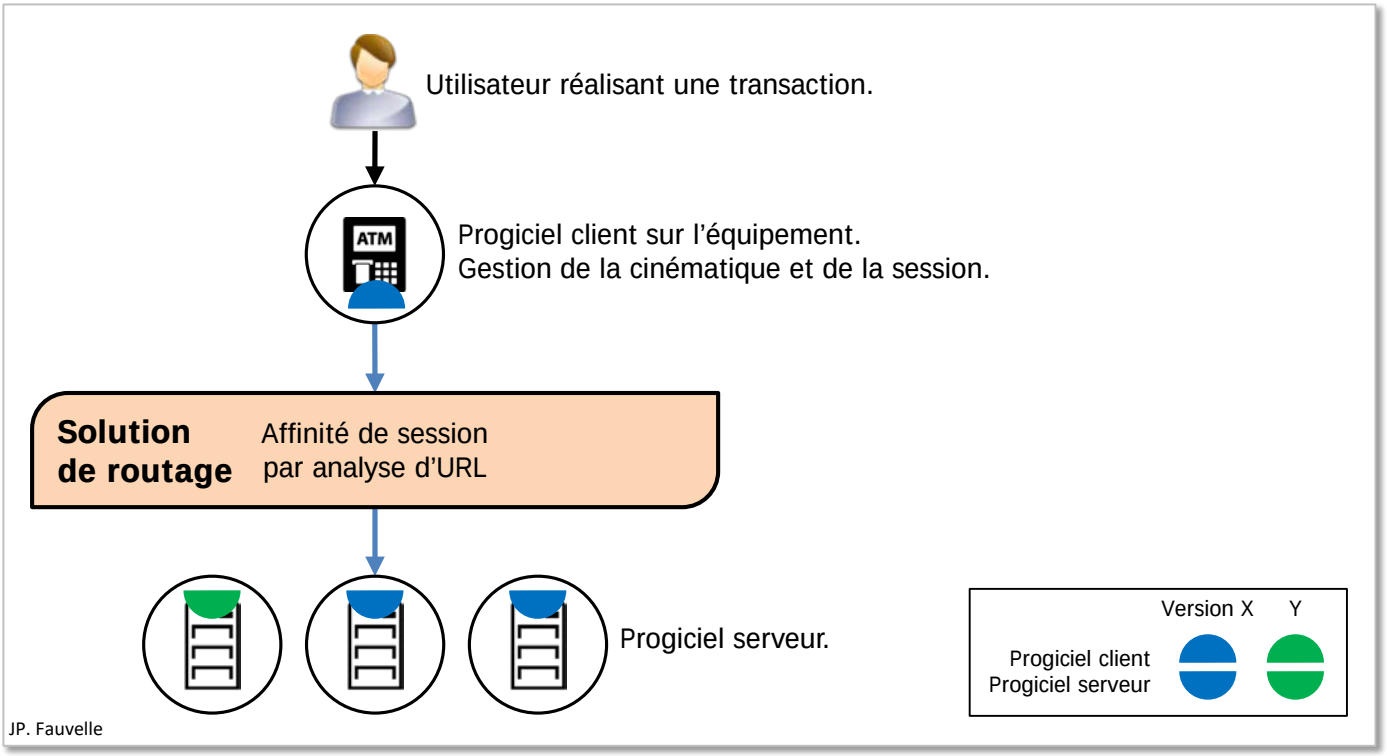

Figure 2 — Illustration des attendus / Intégration sans coutures.

1.3.4 PRISE EN COMPTE DES VERSIONS DU PROGICIEL

Le progiciel installé sur les équipements et les serveurs est probablement déployé dans plusieurs versions distinctes dont la compatibilité est généralement spécifiée sous la forme d'une matrice de compatibilité qui doit être prise en compte.

À cette fin (Figure 3) :

- La solution idéale de routage prend en compte la compatibilité entre les différentes versions du progiciel déployé, de sorte qu'une requête provenant d'une version cliente du progiciel sur un équipement de la flotte, ne soit routée que vers un serveur hébergeant une version compatible du progiciel ;
- La solution découvre automatiquement la version du progiciel par analyse d'URL et/ou via la consultation d'un référentiel d'entreprise, lorsque ce dernier existe.

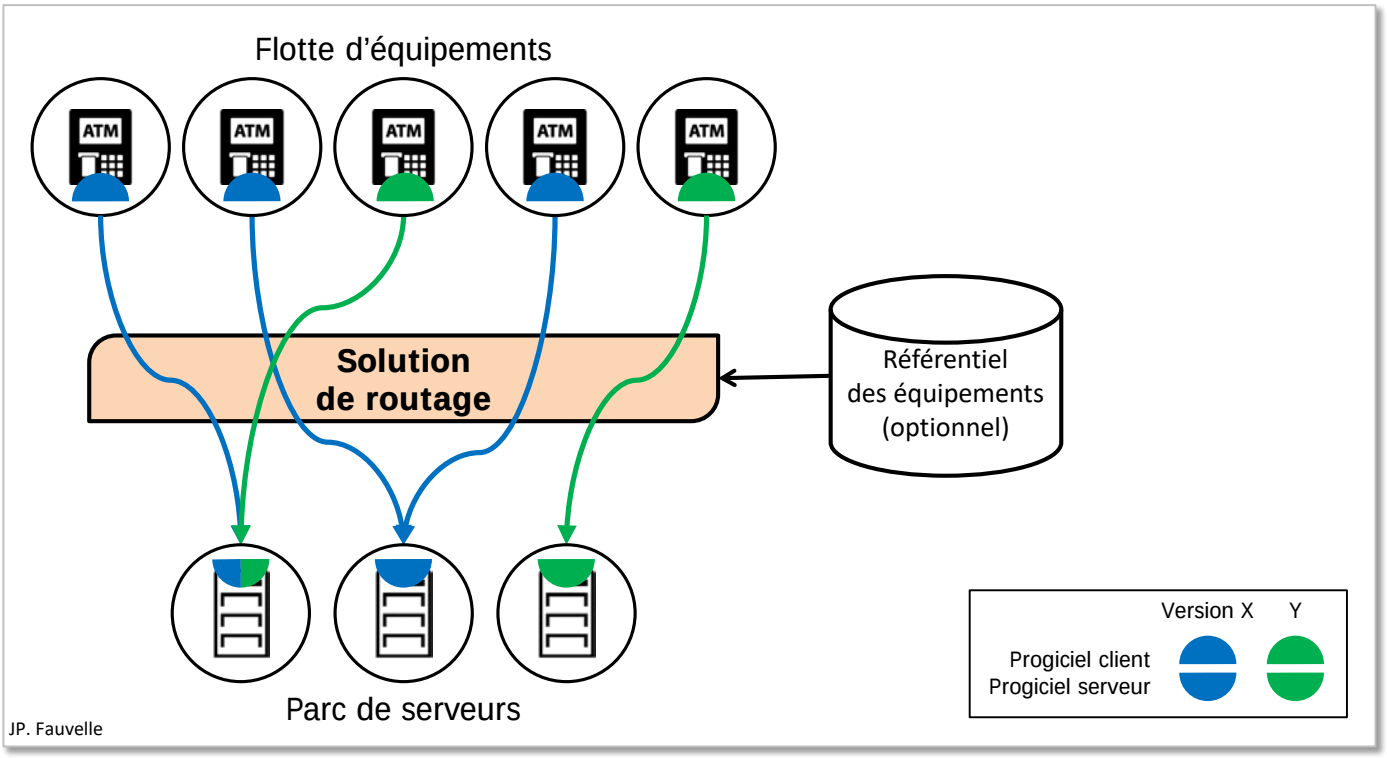

Figure 3 — Illustration des attendus / Prise en compte des versions du progiciel.

1.3.5 ADAPTATION À LA TOPOLOGIE DE LA PLATEFORME

L'architecture de la plateforme existante comprend souvent un découpage en sites ou silos, entre lesquels une synchronisation de données complète ou partielle, immédiate ou différée, peut exister.

Lorsque des flux provenant d'un équipement donné, et initialement routés vers un serveur dit « source », doivent être déroutés vers un serveur dit « cible », plusieurs cas de figure sont possibles lors du choix d'un serveur cible, en fonction de la topologie de la plateforme :

- Certains serveurs ne peuvent jamais être choisis comme cible ;
- Certains serveurs peuvent toujours être choisis comme cible ;
- Certains serveurs peuvent être choisis comme cible mais cela provoquera une indisponibilité temporaire. Ce choix est donc acceptable lorsque un équipement est déjà incidenté, ou lorsque une transaction est terminée (l'équipement ne sera pas disponible quelque temps, mais aucune transaction n'a été interrompue ni aucun utilisateur impacté).

Afin de respecter ces contraintes topologiques (Figure 4) :

- La solution idéale de routage permet de configurer la topologie de la plateforme, avec effet sans interruption de service ;
- La solution permet le déroutage des flux dans le respect de la topologie de la plateforme.

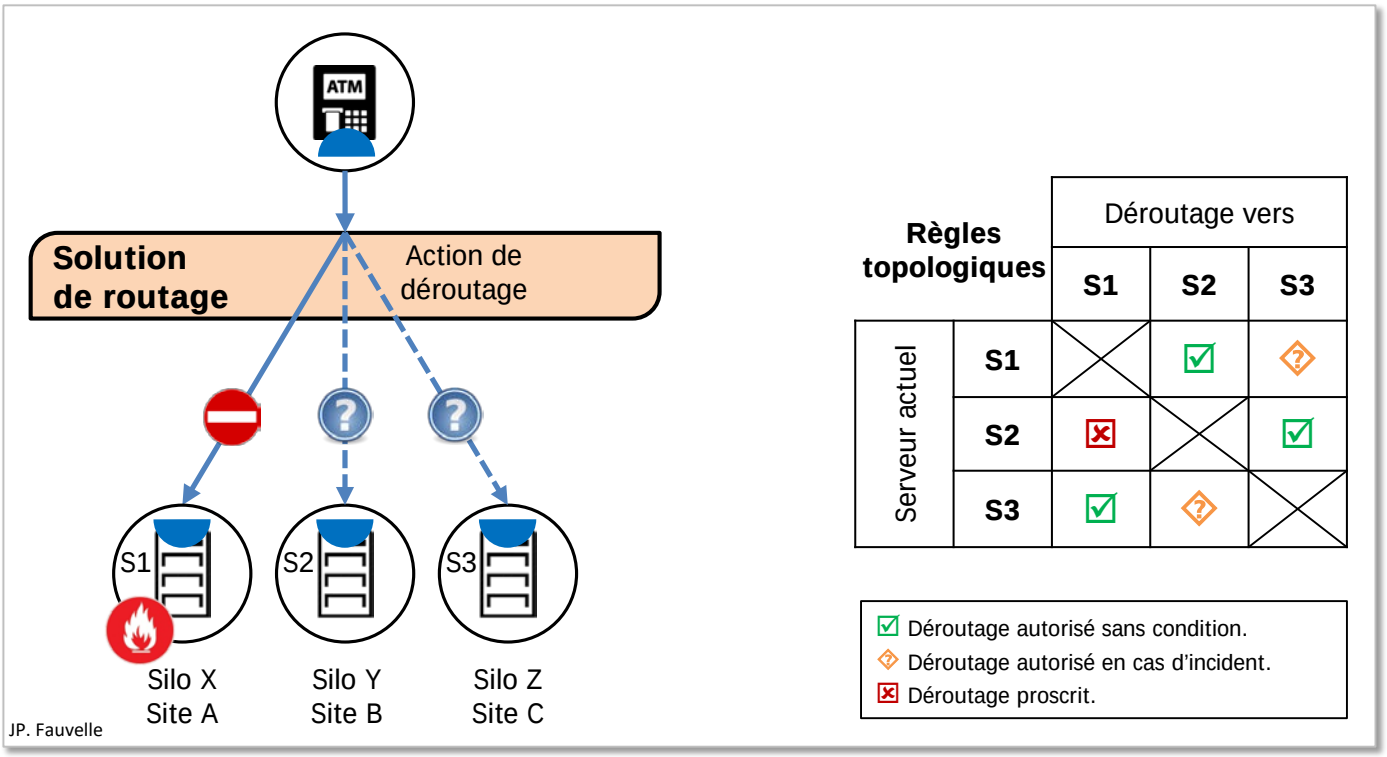

Figure 4 — Illustration des attendus / Adaptation à la topologie de la plateforme.

1.3.6 CAPACITÉ DE MISE À L'ÉCHELLE

Parce que la gestion de capacité est une exigence industrielle forte (Figure 5) :

- La solution idéale de routage offre une capacité de mise à l'échelle horizontale et illimitée ;
- La solution s'adapte à toute volumétrie, toute taille de flotte et de parc ;
- La croissance se fait par ajout de nœuds, avec effet sans interruption de service.

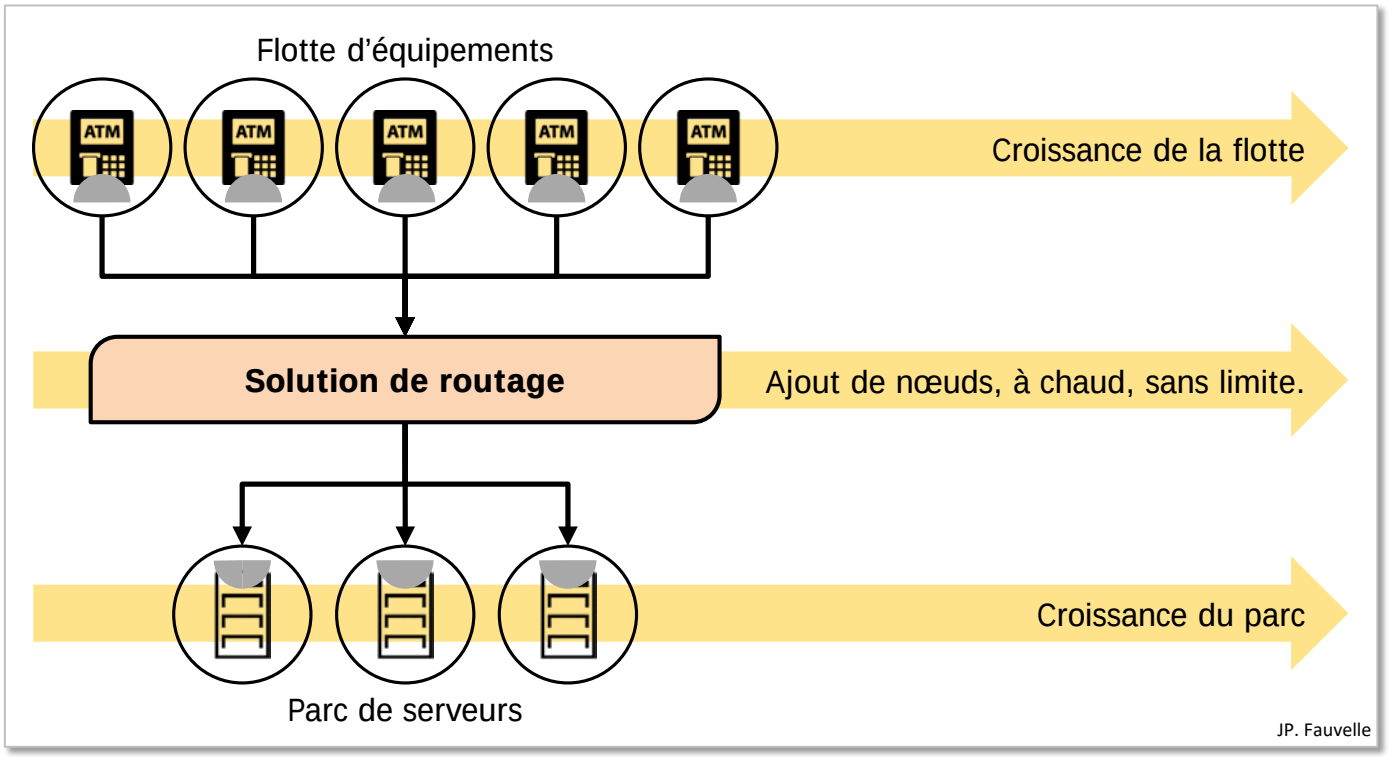

Figure 5 — Illustration des attendus / Capacité de mise à l'échelle.

1.3.7 SEGMENTATION MULTICRITÈRES

La gestion d'une flotte et d'un parc peut nécessiter certains types d'opérations, en ce compris :

- Le cloisonnement des domaines (production, pré-production, développement, test, etc.) ;
- La mise en service progressive et automatique d'un ensemble d'équipements, à un horaire programmé ;
- L'isolation progressive et automatique d'un ensemble d'équipements ou de serveurs, à un horaire programmé (maintenance, tests, etc.) ;
- La segmentation logique de la flotte et du parc (canal, partenaire, catégorie, zone, région, etc.) ;
- L'application de niveaux de service distincts en fonction de critères souhaités.

À cette fin (Figure 6) :

- La solution idéale de routage permet la segmentation des équipements et des serveurs en groupes logiques multicritères d'équipements et de serveurs ;
- La solution permet la gestion de règles d'affinités et d'exclusions entre groupes, avec date d'effet programmable ;
- La solution permet d'inclure un équipement ou un serveur dans plusieurs groupes, les exclusions étant prioritaires sur les affinités.

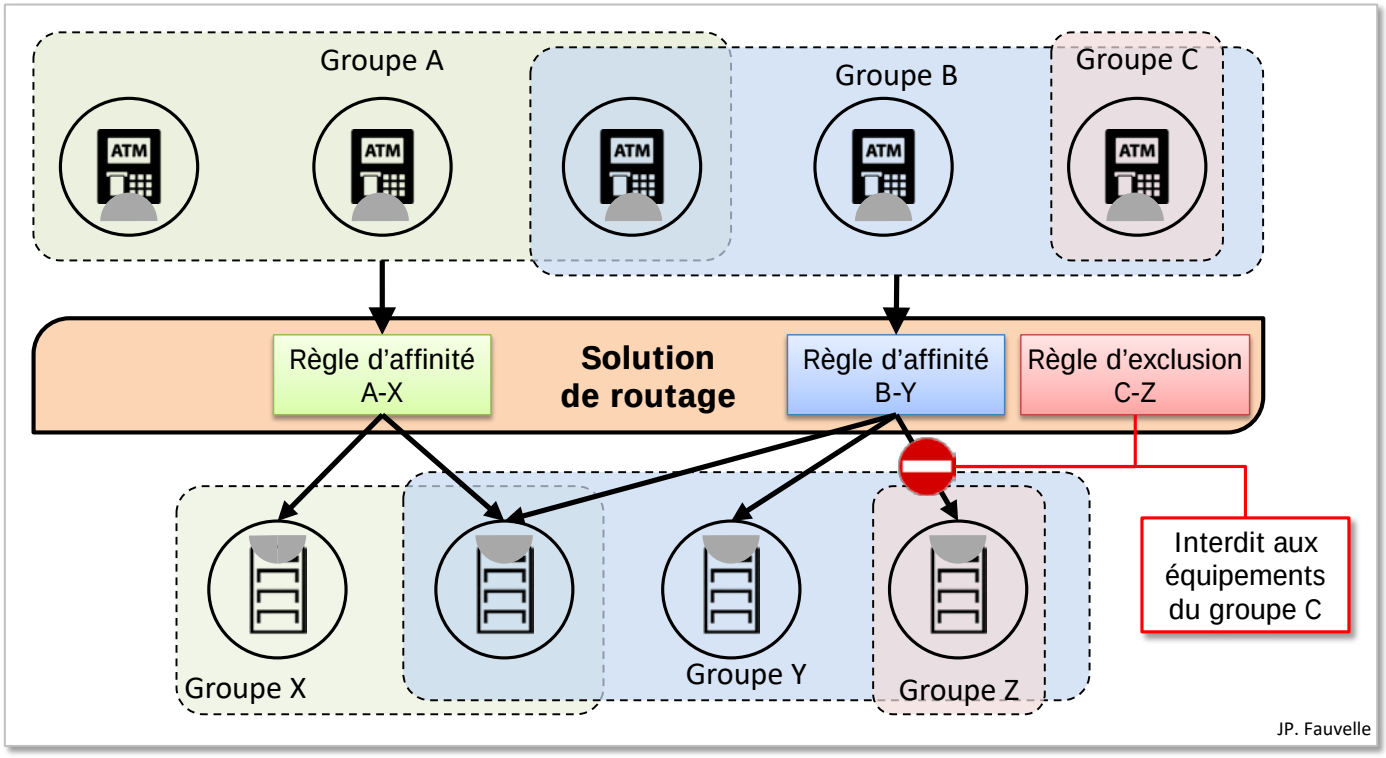

Figure 6 — Illustration des attendus / Segmentation multicritères.

1.3.8 ASSERVISSEMENT DE LA CHARGE ET DU TRANSFERT DE CHARGE

Si l'asservissement de charge est une fonction commune et simple à mettre en œuvre, l'asservissement du transfert de charge est une fonction plus rare et complexe.

En effet, les requêtes déroutées d'un serveur source vers un serveur cible peuvent temporairement consommer sur le serveur cible davantage de ressources informatiques que ces mêmes requêtes lorsqu'elles sont simplement maintenues dans leur route.

Cette surconsommation se retrouve notamment chez certains progiciels devant réaliser des tâches particulières au moment où ils traitent les primo-requêtes émises par les équipements (initialisations, récupération d'un contexte, génération de clés, etc.). Cette surconsommation inhérente au progiciel peut même parfois dépendre de la version du progiciel et varier en fonction de la nature des requêtes.

Une même requête peut donc consommer davantage selon qu'elle est routée (maintien de route) vers le même serveur ou déroutée vers un autre serveur compatible, et dans ce second cas, cette surconsommation engendrée sur le serveur peut dépendre de la version du progiciel et du type de requête. À cela s'ajoute l'éventuelle hétérogénéité du parc dont les serveurs ne sont pas tous d'égale puissance.

À défaut de prendre en considération cette problématique de charge et de transfert de charge, les conséquences peuvent être sérieuses. À titre d'exemple, la perte d'un site « A » rend nécessaire de dérouter d'urgence les flux de nombreux équipements, vers les serveurs d'un site « B ». Même si ce site B possède les ressources informatiques suffisantes pour traiter la charge A + B, le surcoût inhérent au déroutage pourrait saturer le site B et provoquer un nouvel incident, provoquant le transfert en masse vers un troisième site C, et ainsi de suite.

Il est donc nécessaire de quantifier cette surcharge de transfert en fonction du progiciel et de sa version ainsi que du type de requête, afin de dérouter les flux des équipements concernés dans un ordre et avec un débit qui maximisent la célérité du transfert sans franchir le seuil de saturation, quitte à sacrifier certains flux, le moins possible et pour la plus courte durée possible.

À cette fin (Figure 7 et Figure 8) :

- La solution idéale de routage prend en compte l'hétérogénéité des serveurs, ainsi que la variabilité de consommation des ressources en fonction du progiciel, de sa version, et du type des requêtes ;
- La solution met en œuvre un double mécanisme quantifiant et asservissant la charge ainsi que le transfert de charge, ce qui évite la propagation en cascade des incidents par effet domino ;
- La solution peut refuser temporairement les requêtes de certains équipements pour maintenir en service le plus grand nombre ;
- La solution améliore le taux de disponibilité de la flotte et optimise l'utilisation des ressources informatiques du parc.

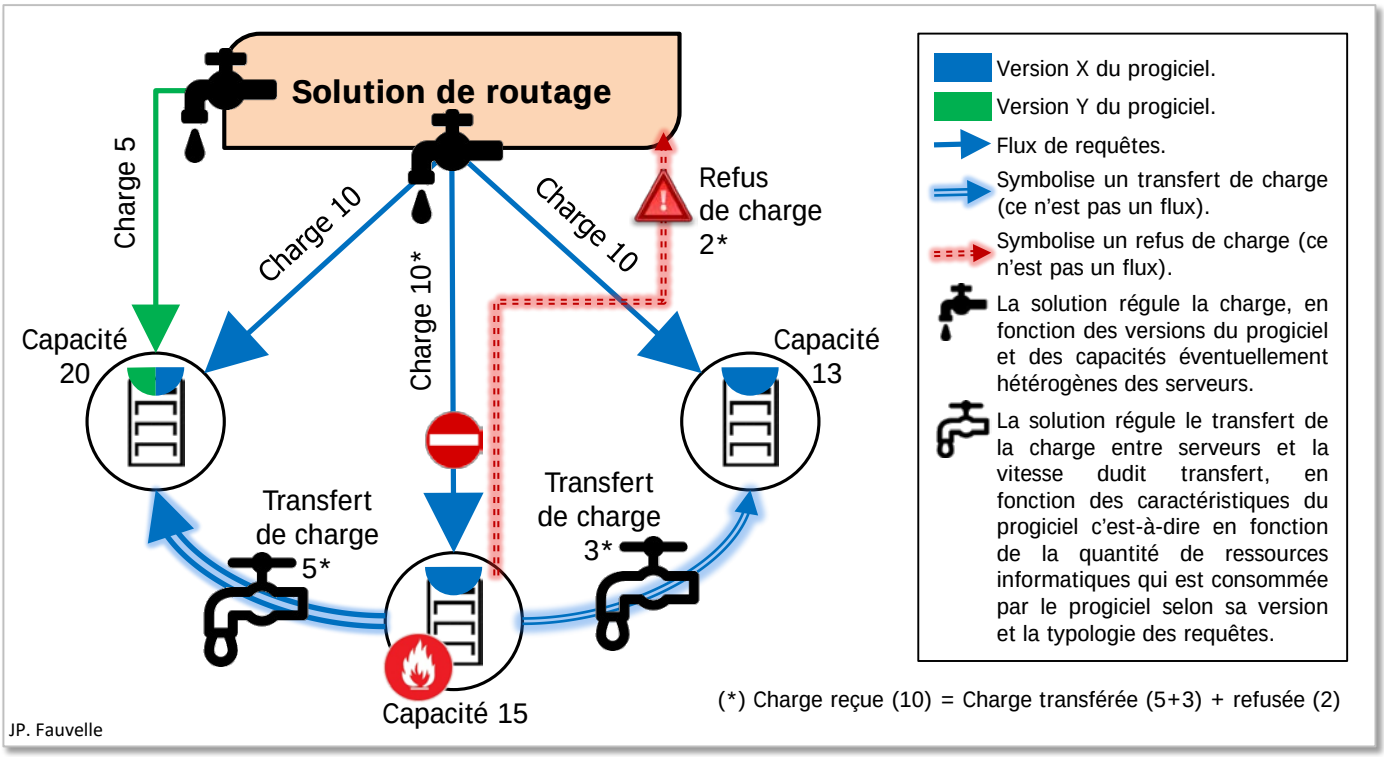

Figure 7 — Illustration des attendus / Asservissement de la charge et du transfert de charge.

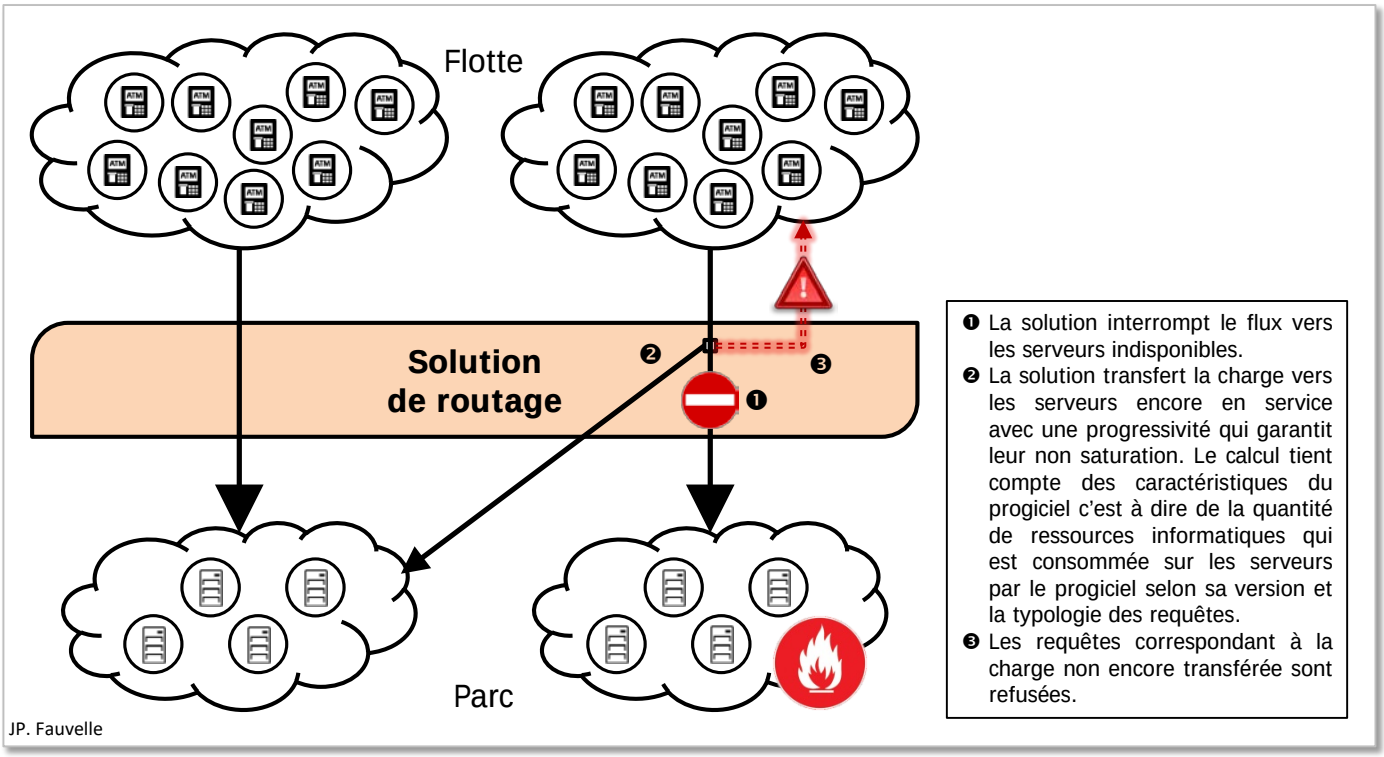

Figure 8 — Illustration des attendus / Contention des incidents en cascade.

1.3.9 INJECTION MARKETING SUR LES ÉQUIPEMENTS

Positionnée entre la flotte et le parc, la solution idéale peut aisément apporter une fonction d'injection marketing sur les équipements, en intervenant dynamiquement au niveau de l'URL sans qu'il soit nécessaire de modifier les équipements (Figure 9) :

- Injection marketing – Affichage d'un contenu différencié sur les équipements ;
- Fonctionne sur toutes les ressources statiques (image, vidéo, son, feuille de style, etc.) ;
- Ne requiert ni installation ni paramétrage sur les équipements, ni référentiel entreprise ;
- Mise en place simple, aucun impact sur la performance.

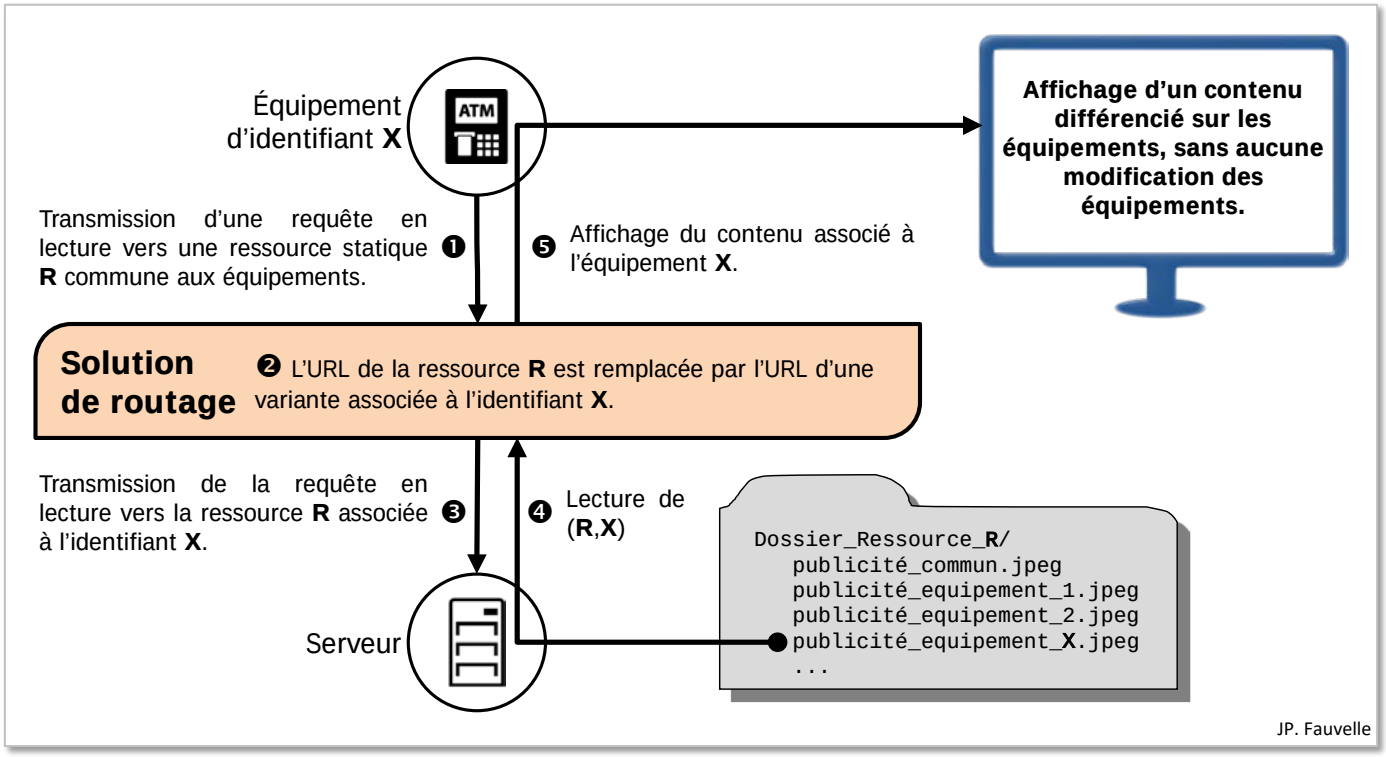

Figure 9 — Illustration des attendus / Injection marketing sur les équipements.

2 PRÉSENTATION DE LA SOLUTION INVENTIVE

2.1 DESCRIPTION DE L'INVENTION

2.1.1 RÉPONSES DE L'INVENTION À LA PROBLÉMATIQUE

Réf.	Caractéristiques de la solution inventive
1	Agit en mandataire inverse (*reverse proxy*) invisible entre les automates et les serveurs, ce qui facilite l'intégration de la solution entre les automates et les serveurs.
2	S'adapte facilement à tout progiciel client / serveur via portage d'un unique point de fonction inter-opérant au niveau session et cinématique du parcours des utilisateurs des automates. Par exemple l'adaptation au protocole HTTP(S) utilisé par les automates se limite à l'analyse des URI et des champs de l'en-tête des requêtes. Ce caractère quasi-agnostique vis à vis du progiciel est un gage d'adaptabilité et d'évolutivité.
3	S'adapte à tout parc de serveurs via paramétrage de règles topologiques intégrant les spécificités du progiciel dans l'architecture en place, ce qui facilite l'intégration dans toute plateforme informatique.
4	Accepte un nombre illimité d'automates grâce à une capacité de mise à l'échelle horizontale à chaud, ce qui simplifie la gestion de capacité, la montée en charge et l'exploitation, tout en réduisant les coûts.
5	Supporte une segmentation logique multicritères du parc (domaine, canal, partenaire, catégorie, versions, géolocalisation / région, etc.) via paramétrage de règles d'affinités et de règles d'exclusions entre des groupes quelconques d'automates et des groupes quelconques de serveurs, ce qui facilite l'exploitation.
6	Prend en compte la multiplicité des versions déployées du progiciel entre automates et serveurs, ce qui facilite l'exploitation et le déploiement.
7	Permet l'enrôlement immédiat des automates sans besoin d'un référentiel optionnel des automates, ce qui facilite l'exploitation et le déploiement.
8	Améliore le taux de disponibilité de la flotte d'automates via une heuristique socio-mimétique, ce qui augmente globalement la qualité de service.
9	Optimise l'utilisation du parc de serveurs en distribuant et plafonnant dynamiquement la charge, ce qui augmente la qualité de service et contribue à réduire les coûts.
10	Optimise l'utilisation du parc de serveurs en régulant la translation de charge et en intégrant l'éventuelle surconsommation de ressources par le progiciel lors du traitement des primo-requêtes post-déroutage (initialisation, échanges de clés de chiffrement, etc.) ce qui augmente la qualité de service.
11	Contient puis régule la propagation cascadée (effet dominos) des surcharges liées aux incidents en masse, ce qui augmente la qualité de service.
12	Permet une vision en temps réel de l'état de la flotte d'automates, du parc de serveurs, des déroutages en cours et de leurs causes, ce qui facilite l'exploitation.
13	Une évolution simple permettrait d'intégrer une fonction dite « d'injection marketing » capable de modifier à la volée certaines requêtes selon l'identifiant de l'automate. À titre d'exemple, cela permettrait d'afficher sur chaque automate un visuel différencié comme des informations sur les commerces de proximité autour de l'automate, sans qu'aucune action différenciée ne soit nécessaire sur la flotte d'automates ni en particulier sur le progiciel client et serveur.

Tableau 2 — Réponses de la solution inventive à la problématique et avantages associés.

2.1.2 MOTEUR DE DONNÉES

L'invention s'appuie notamment sur un moteur de type IMDBMS (in-memory database management system) RAM-Cloud NoSQL temps réel, également nommé simplement « RAM-Cloud » dans la suite du document[4].

Toutefois, la présente invention de routage pourrait s'appuyer sur un autre moteur dès lors que ce dernier permet d'accéder aux informations sous la forme de tables, chaque table étant composée de paires, chaque paire étant notée `{ clé = (liste des champs composant la clé) ; données = (liste des champs constituant les données associées à la clé) }` ou plus simplement `{ clé ; données }`.

Par souci de clarté, le présent document ne décrit l'invention de routage que dans le cadre d'une utilisation avec ledit RAM-Cloud.

2.1.3 PRINCIPE DE L'INVENTION

L'invention est un procédé technique comprenant d'une part des logiciels et d'autre part une pluralité de serveurs informatiques destinés à exécuter lesdits logiciels.

L'invention s'insère dans le système d'information comprenant le parc de serveurs. Elle implémente un routage applicatif intelligent des flux entre ladite flotte et ledit parc (Figure 10 et Figure 11 / Flux 1 et 2).

Une couche de routage technique simple, généralement composée d'équipements standards, assure la transmission du flux depuis la flotte vers la solution. Cette couche ne comporte aucun procédé inventif.

L'invention comprend trois couches inter-opérantes, nommées « Front », « Middle » et « Back » (Figure 10 et Figure 11) :

- La couche Front comprend une pluralité de serveurs informatiques qui lui sont dédiés, ainsi que des logiciels dont le plus important implémente une fonction nommée F_ROUTE. Cette dernière réalise le routage applicatif des flux en exécutant les ordres qui lui sont communiqués par la fonction F_COMPIL de la couche Middle.

  En d'autres termes, F_ROUTE est **l'exécutant** du routage des flux.

- La couche Middle comprend une pluralité de serveurs informatiques qui lui sont dédiés, ainsi que des logiciels dont le plus important implémente une fonction nommée F_COMPIL. Cette dernière implémente l'intelligence du routage en compilant des ordres de déroutages également nommés déroutages ou routes, destinés à être exécutés par la fonction F_ROUTE de la couche Front.

  En d'autres termes, F_COMPIL est le **donneur d'ordre** du routage des flux.

- La couche Back comprend une pluralité de serveurs informatiques qui lui sont dédiés, ainsi que des logiciels implémentant les IHM (Interface Homme-Machine) permettant d'utiliser l'invention.

Les procédés techniques inventifs sont essentiellement localisés au sein des fonctions F_ROUTE et F_COMPIL de la solution.

[4] Jean-Philippe Fauvelle. A distributed, probabilistic, hysteretic, retroactive algorithm to locally weaken the CAP/Brewer's theorem when reading data on a distributed computer system. 2020. ⟨hal-01966396⟩. https://hal.archives-ouvertes.fr/hal-01966396

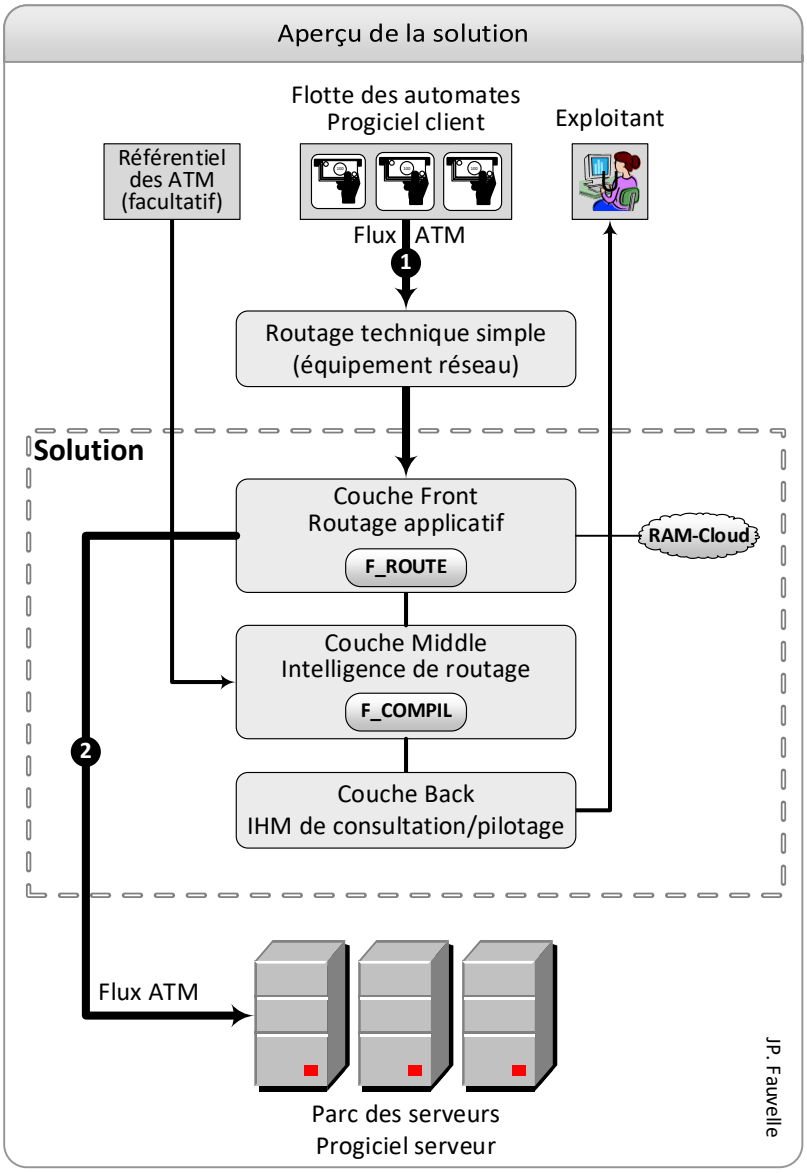

Figure 10 — Aperçu de la solution inventive.

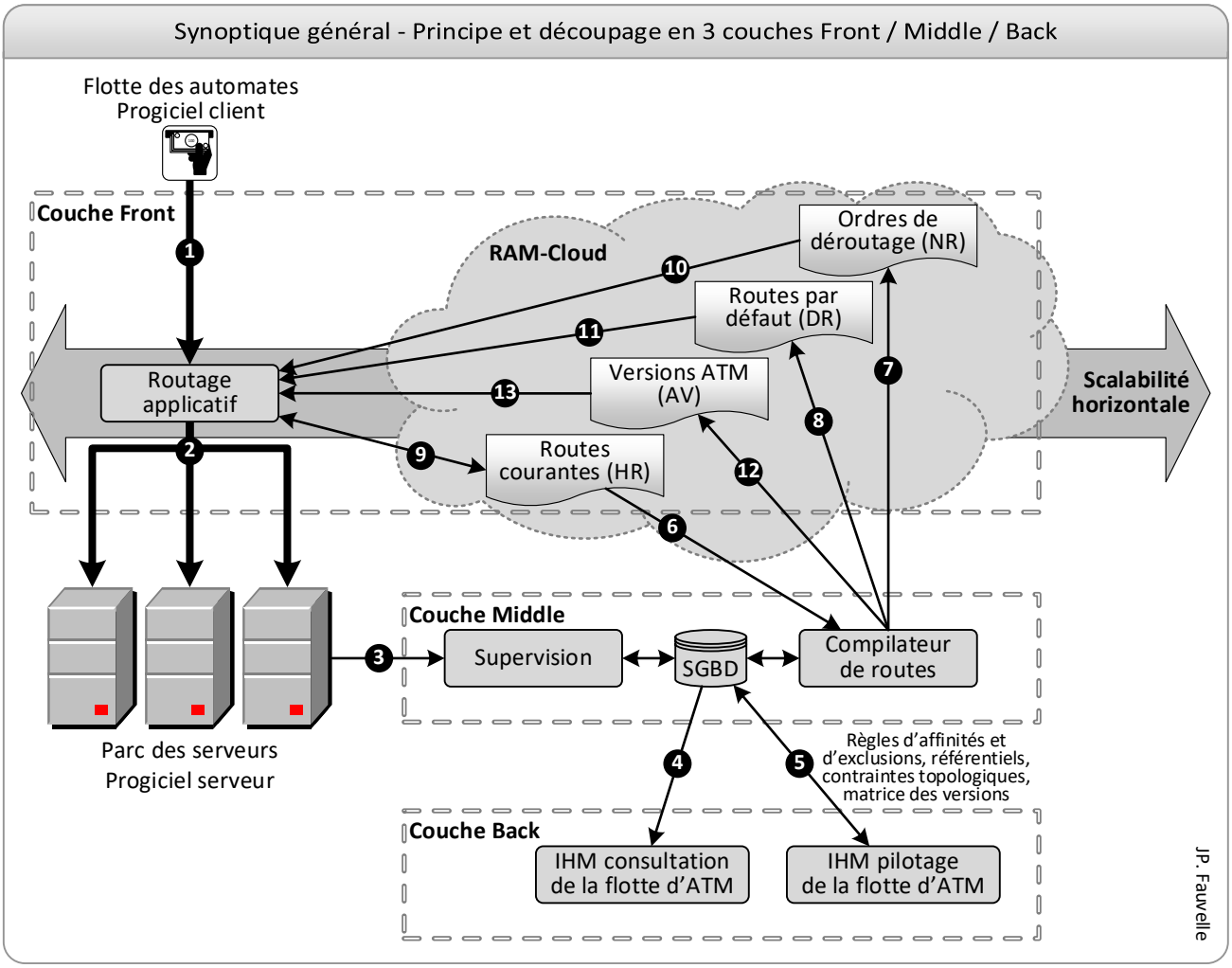

Figure 11 — Synoptique général, principe et découpage en 3 couches.

2.1.4 COUCHE « BACK » : IHM

La couche Back, bien qu'opérant des services de haut niveau dans lesquels réside la valeur ajoutée, est peu innovante car elle n'offre in fine qu'une IHM s'appuyant sur les fonctions implémentées dans les couches Middle et Front.

L'IHM de la couche Back implémente les fonctions suivantes :

A. Afficher un état détaillé, instantané et avec historique, portant notamment sur la flotte d'automates et le parc de serveurs, les règles et leurs effets, les routages en cours, les déroutages à venir et leurs causes (incident, règle, régulation de charge, etc.), les incidents et indisponibilités, l'état de l'équilibrage de charge, etc.

   Toutes ces informations peuvent servir à alimenter un système analytique tiers.

B. Fournir des fonctions de gestion de la flotte d'automates et du parc de serveurs, unitaire ou par lots d'automates / serveurs, en particulier :

   1. Gestion du référentiel de **compatibilité** des multiples versions du progiciel client et du progiciel serveur.
   2. Gestion du référentiel des **serveurs** et de la suspension volontaire desdits serveurs.
   3. Gestion du référentiel des **routes par défaut** et doublement par défaut.

4. Gestion des **règles topologiques** du parc de serveurs. Ces règles définissent, pour chaque couple (serveur S1 source, serveur S2 cible), la condition dans laquelle tout automate est autorisé à être dérouté de S1 vers S2.

   Les 3 conditions possibles sont :

   a. Condition C_JAMAIS : le déroutage est invariablement proscrit de S1 vers S2.
   b. Condition C_INCIDENT : le déroutage de S1 vers S2 n'est autorisé que pour faire face à un incident sur le serveur source S1.
   c. Condition C_TOUJOURS : le déroutage est autorisé sans condition de S1 vers S2.

5. Gestion des **règles d'affinités** entre des groupes d'automates (*) et des groupes de serveurs, lesdits groupes résultant d'une sélection sur des critères multiples (domaine, canal, partenaire, catégorie, versions, géolocalisation, région, etc.) :

   a. Lorsqu'une règle d'affinité R définit un groupe d'automates comprenant un automate A, et lorsque R est active, alors l'automate A est dit **affecté** par la règle R.
   b. Lorsqu'un automate A est affecté par une unique règle d'affinité R, alors A ne peut être routé que vers un serveur compris dans le groupe de serveurs défini par R.
   c. Lorsqu'un automate A est affecté par plusieurs règles d'affinités, alors A ne peut être routé que vers un serveur compris dans le groupe de serveurs défini par l'une au moins desdites règles.
   d. Lorsqu'un automate A n'est affecté par aucune règle d'affinité, alors A peut être routé vers n'importe lequel des serveurs existant. En d'autres termes, l'absence de règles revêt un caractère permissif.
   e. Les exclusions sont prioritaires sur les affinités (**).

6. Gestion des **règles d'exclusions** entre des groupes d'automates (*) et des groupes de serveurs, lesdits groupes résultant d'une sélection sur des critères multiples (domaine, canal, partenaire, catégorie, versions, géolocalisation, région, etc.) :

   a. Lorsqu'une règle d'exclusion R définit un groupe d'automates comprenant un automate A, et lorsque R est active, alors l'automate A est dit **affecté** par la règle R.
   b. Lorsqu'un automate A est affecté par une unique règle d'exclusion R, alors A ne peut pas être routé vers un serveur compris dans le groupe de serveurs défini par R.
   c. Lorsqu'un automate A est affecté par plusieurs règles d'exclusion, alors A ne peut pas être routé vers un serveur compris dans le groupe de serveurs défini par l'une au moins desdites règles.
   d. Lorsqu'un automate A n'est affecté par aucune règle d'exclusion, alors A peut être routé vers n'importe lequel des serveurs existant. En d'autres termes, l'absence de règles revêt un caractère permissif.
   e. Les exclusions sont prioritaires sur les affinités (**).

(*) *Les automates pouvant être affectés à des groupes sont d'une part les automates visibles en raison de leur activité (c.-à-d. les automates enrôlés complètement ou partiellement) et d'autre part les automates présents dans un référentiel appartenant à l'entreprise. Ledit référentiel, facultatif, permet d'inclure des automates dans les groupes des règles afin d'anticiper les premières requêtes desdits automates. Ce procédé permet d'administrer le comportement du routage intelligent appliqué à tous les automates y compris ceux qui ne sont pas encore en service.*

(**) *Lorsqu'un automate est routable vers un serveur sous l'effet d'une règle d'affinité, et simultanément est interdit de routage vers ledit serveur sous l'effet d'une règle d'exclusion, alors l'interdiction l'emporte.*

2.1.5 COUCHE « FRONT » (F_ROUTE)

2.1.5.1 MANDATAIRE INVERSE

La fonction F_ROUTE reçoit les requêtes émises par le progiciel équipant les automates et les transmet de manière transparente vers les serveurs où elles sont traitées par le progiciel équipant lesdits serveurs, ledit progiciel émettant alors une réponse qui effectue le cheminement inverse jusqu'aux automates.

La transmission technique des requêtes repose sur un mandataire inverse (*reverse proxy*) asservi par la fonction F_ROUTE, cette dernière indiquant audit mandataire la destination de routage de chaque requête.

La fonction F_ROUTE délègue ledit rôle de mandataire inverse à un composant du marché (exemple : le produit open-source Apache et son module « mod_proxy »).

2.1.5.2 RELATION ENTRE SERVEUR, ROUTE, ADRESSE ET PORT

Une destination de routage, également nommée « route », est constituée d'un couple (SRV_ADDR, SRV_PORT) également noté « SRV_ADDR:SRV_PORT », où SRV_ADDR est l'adresse du serveur de traitement des requêtes et SRV_PORT est le port d'écoute du progiciel sur ledit serveur. L'adresse du serveur peut être son nom d'hôte (*hostname*) ou son adresse IP.

Il est techniquement possible que sur un serveur donné, le progiciel soit à l'écoute des requêtes sur plusieurs adresses et plusieurs ports. Dans un tel cas, plusieurs routes mènent vers un même serveur.

Toutefois, cette possibilité n'offrant aucun intérêt pour l'invention, il est considéré dans la suite du présent document que le progiciel, sur chaque serveur, est à l'écoute sur un unique couple (SRV_ADDR, SRV_PORT).

Dans cette configuration, il existe une bijection entre une route et un serveur. En d'autres termes, une route notée (SRV_ADDR, SRV_PORT) ou « SRV_ADDR:SRV_PORT » est associée à un unique serveur identifié par SRV_ADDR, et un serveur identifié par SRV_ADDR est associé à une unique route notée (SRV_ADDR, SRV_PORT) ou « SRV_ADDR:SRV_PORT ».

2.1.5.3 COLLECTE DES CHAMPS ET ADHÉRENCE AVEC LE PROGICIEL

À chaque requête HTTP(S) reçue, notée Q, la fonction F_ROUTE en extrait les informations suivantes :

1. L'adresse IP de l'automate (Q.atm_ip) qui peut changer au cours du temps mais reste constante au cours d'une session. Cette information est toujours disponible car collectée par F_ROUTE via le système d'exploitation.
2. L'horodatage de la requête (Q.req_time) précis au moins à la milliseconde. Cette information est toujours disponible car mesurée par F_ROUTE via le système d'exploitation.
3. L'identifiant de l'automate (Q.atm_id) transmis dans l'URI de la requête ou dans un champ d'en-tête HTTP(S). Selon les capacités du progiciel, cette information n'est pas toujours transmise dans chaque requête, auquel cas le dernier identifiant connu de l'automate est utilisé.

   Ce dernier identifiant connu est mémorisé dans le champ HR.atm_id de la route courante HR, cette dernière étant associée via sa clé primaire HR.atm_ip à l'adresse IP toujours disponible de l'automate (Tableau 6).

4. La version du progiciel de l'automate (Q.atm_version). Selon les capacités du progiciel, cette information n'est pas toujours transmise dans chaque requête, auquel cas la dernière version connue associée à l'adresse IP de l'automate est utilisée.

   Cette dernière version connue est mémorisée dans le champ HR.atm_version de la route courante HR, cette dernière étant associée via sa clé primaire HR.atm_ip à l'adresse IP toujours disponible de l'automate (Tableau 6).

5. Le caractère Q.is_eos de la requête (*is end of session*) indiquant si cette requête caractérise ou non la fin de la session du porteur de carte utilisant l'automate. Cette information est déterminée par F_ROUTE simplement en analysant la forme de l'URI de la requête reçue.

6. Le caractère Q.is_ping de la requête, indiquant que la requête est de type *ping* c'est à dire qu'elle constitue une surveillance par l'automate de l'état du serveur qui traite ses requêtes, par opposition aux requêtes de type métier correspondant à l'activité d'un porteur de carte sur l'automate. Une requête est soit de type métier soit de type *ping*.

   Cette information permet de différencier un automate simplement vivant d'un automate actif c.-à-d. utilisé. Cette information non indispensable est déterminée par F_ROUTE simplement en analysant la forme de l'URI de la requête reçue.

Il convient de souligner que le corps de la requête émise par le progiciel client vers le progiciel serveur ainsi que le corps de la réponse émise du progiciel serveur vers le progiciel client, ne sont jamais lus ni modifiés par F_ROUTE.

Ainsi, l'ensemble de ces champs peut être collecté simplement et sans créer d'adhérence forte ni avec le progiciel client ni avec le progiciel serveur.

De plus, l'adhérence avec le progiciel est concentrée sur un unique point de fonction au sein de F_ROUTE, ce qui facilite l'adaptation de l'invention à tout progiciel.

2.1.5.4 CINÉMATIQUE SIMPLIFIÉE D'ACCÈS AUX DONNÉES

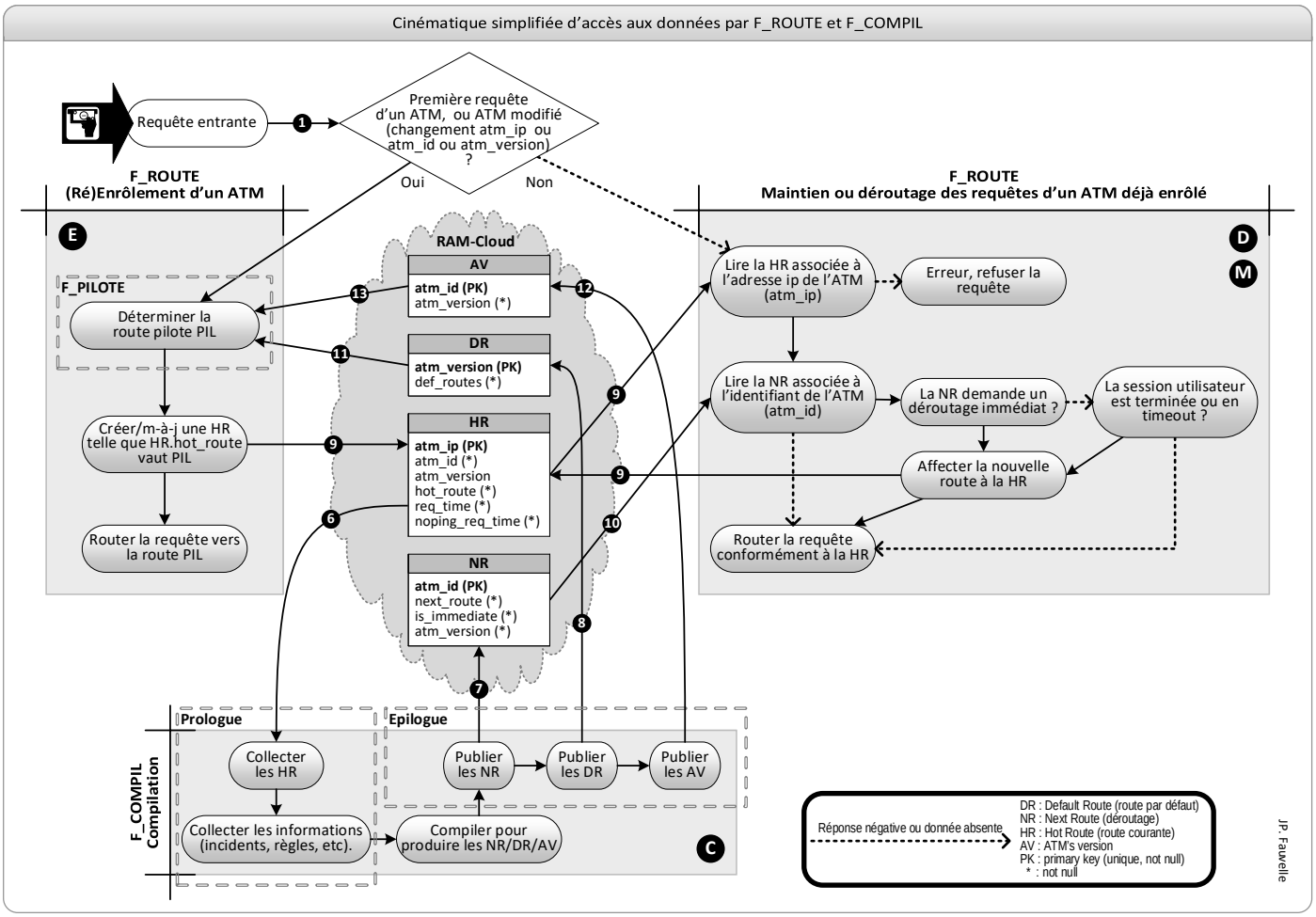

Figure 12 — Cinématique simplifiée d'accès aux données.

2.1.5.5 MODÈLE DE DONNÉES (AV, DR, NR, HR)

La cinématique simplifiée d'accès aux données est schématisée en Figure 12.

Les 4 types de données (AV, DR, NR, HR) sont décrits ci-après.

2.1.5.5.1 VERSIONS PROGICIELLES DES AUTOMATES / ATM'S VERSIONS (AV)

Les données AV sont produites par la fonction de compilation depuis la couche Middle qui les écrit dans le RAM-Cloud, puis elles sont consultées par la fonction de routage depuis la couche Front (Figure 11 et Figure 12 / Flux 12 et 13).

La fonction de compilation obtient ces données AV auprès du référentiel des automates de l'entreprise, facultatif. Il est donc possible, selon la complétude dudit référentiel, s'il existe, que lesdites AV soit inexistantes ou ne couvrent que partiellement ladite flotte.

Les données AV sont persistées dans le RAM-Cloud sous la forme d'une table constituée de paires dont chacune est notée `{ clé = (atm_id) ; données = (atm_version) }` conformément à ce schéma :

Champ	Propriété	Description
AV.atm_id	Clé primaire	Identifiant de l'automate.
AV.atm_version	Obligatoire	Version du progiciel automate, provenant du référentiel des automates de l'entreprise.

Tableau 3 — Format de la table des versions automates (AV).

En certaines occasions, la fonction de routage requiert de connaitre la version du progiciel d'un automate A d'identifiant ID qui est à l'origine de la requête Q dont le routage est en cours.

Si cette information requise n'est pas déjà mémorisée dans le champ HR.atm_version de la route courante HR si elle existe, et n'est pas non plus transmise dans le champ Q.atm_version de la requête Q, alors elle peut être lue dans le champ AV.atm_version de la paire AV dont la clé AV.atm_id est égale à ID.

2.1.5.5.2 ROUTES PILOTES / ROUTES PAR DÉFAUT / DEFAULT ROUTES (DR)

Les routes par défaut également nommées « routes pilotes » ou DR sont produites par la fonction de compilation depuis la couche Middle qui les écrit dans le RAM-Cloud, puis elles sont consultées par la fonction de routage depuis la couche Front (Figure 11 et Figure 12 / Flux 8 et 11).

Les routes par défaut sont persistées dans le RAM-Cloud sous la forme d'une table constituée de paires dont chacune est notée `{ clé = (atm_version) ; données = (def_routes) }` conformément à ce schéma :

Champ	Propriété	Description	Exemple de valeur
DR.atm_version	Clé primaire	Soit la version du progiciel automate. Soit « MOST_COMMON_VERSION ».	
DR.def_routes	Obligatoire	Liste de toutes les routes disponibles et compatibles avec DR.atm_version.	« Host_1:8001, Host_4:9005, Host_6:7074 »

Tableau 4 — Format de la table des routes par défaut (DR).

Dans ladite table, à chaque version DR.atm_version du progiciel automate, est associée la liste DR.def_routes comprenant toutes les routes qui sont compatibles avec DR.atm_version (*) et disponibles (**) au moment où les routes par défaut sont produites.

(*) *Une route notée (SRV_ADDR, SRV_PORT) ou « SRV_ADDR:SRV_PORT » est dite compatible avec un automate lorsque la version du progiciel serveur équipant le serveur SRV_ADDR est compatible avec la version du progiciel client équipant ledit automate.*

(**) *Un serveur est dit disponible lorsque il n'est ni incidenté ni suspendu volontairement.*

En outre, ladite table peut éventuellement comporter une paire particulière dont la clé DR.atm_version est égale à la constante « MOST_COMMON_VERSION » et qui se comporte comme une route doublement par défaut, dans laquelle le champ DR.def_routes contient la liste des routes disponibles dont la version progicielle est la plus **commune** du parc de serveurs c.-à-d. la plus **probablement** compatible (sans certitude) avec le plus grand nombre d'automates.

Cette route **doublement par défaut** est une sécurité qui garantit l'attribution d'une route statique en toute circonstance.

En certaines occasions, la fonction de routage doit attribuer une route statique par défaut à un automate A, en exécutant le mode opératoire suivant :

1. Si la version V du progiciel client équipant l'automate A est connue, c.-à-d. lue en première intention dans Q.atm_version ou à défaut dans HR.atm_version :
    a. S'il existe une paire DR telle que sa clé DR.atm_version est égale à V, alors une route par défaut **assurément compatible** avec le progiciel équipant A peut être choisie parmi celles listées dans le champ DR.def_routes.

b. Dans le cas contraire, le mode opératoire se poursuit au point 2a ci-après.

2. Dans le cas contraire (c.-à-d. si la version du progiciel de A n'est pas connue) :
   a. S'il existe une paire DR telle que sa clé DR.atm_version est égale à la constante « MOST_COMMON_VERSION » alors une route doublement par défaut **probablement compatible** avec le progiciel de A peut être choisie parmi celles listées dans le champ DR.def_routes.
   b. Dans le cas contraire, il n'est pas possible d'attribuer une route statique par défaut en raison d'une insuffisance de paramétrage.

Ce mode opératoire est détaillé au §2.1.5.6.10.

2.1.5.5.3 ORDRES DE DÉROUTAGE / NEXT ROUTES (NR)

Les ordres de déroutage également nommés « déroutages » ou NR, sont produits par la fonction de compilation depuis la couche Middle qui les écrit dans le RAM-Cloud, puis ils sont consultés par la fonction de routage depuis la couche Front (Figure 11 et Figure 12 / Flux 7 et 10).

Les déroutages sont persistés dans le RAM-Cloud sous la forme d'une table constituée de paires, dont chacune est notée `{ clé = (atm_id) ; données = (next_route, is_immediate, atm_version) }` conformément à ce schéma :

Champ	Propriété	Description	Exemple de valeur
NR.atm_id	Clé primaire	Identifiant de l'automate.	
NR.next_route	Obligatoire	Spécifie soit la nouvelle route à suivre au format « SRV_ADDR:SRV_PORT », soit la constante « REFUSED » lorsque les requêtes provenant de cet automate doivent être refusées.	« Host_1:8001 » « REFUSED »
NR.is_immediate	Obligatoire	Spécifie le caractère d'immédiateté de l'ordre de déroutage.	VRAI FAUX
NR.atm_version	Obligatoire	Version du progiciel client équipant l'automate. La valeur est la copie du champ HR.atm_version de la route courante HR préalablement lue par le compilateur en phase prologue pour ledit automate (Figure 12 / Flux 6). Comme le compilateur ne peut dérouter que les automates dont l'enrôlement est complet (c.-à-d. dont le champ HR.atm_version est défini) alors le champ NR.atm_version est nécessairement défini lui aussi.	

Tableau 5 — Format de la table des ordres de déroutages (NR).

Les ordres de déroutages produits par la fonction de compilation constituent le carnet d'ordres exécuté par la fonction de routage.

2.1.5.5.4 ROUTES COURANTES / HOT ROUTES (HR)

Les routes courantes également nommés HR, sont produites et consultées par la fonction de routage depuis la couche Front, elles sont également consultées par la fonction de compilation depuis la couche Middle (Figure 11 et Figure 12 / Flux 6 et 9).

Les routes courantes sont persistées dans le RAM-Cloud sous la forme d'une table constituée de paires, dont chacune est notée `{ clé = (atm_ip) ; données = (atm_id, atm_version, hot_route, req_time, noping_req_time) }` conformément à ce schéma :

Champ	Propriété	Description	Exemple de valeur
HR.atm_ip	Clé primaire	Adresse IP de l'automate.	
HR.atm_id	Obligatoire	Identifiant de l'automate.	
HR.atm_version	Facultatif	Version du progiciel client équipant l'automate. L'enrôlement de l'automate est dit **complet** lorsque le champ atm_version de sa route courante est défini, **partiel** ou **incomplet** dans le cas contraire.	
HR.hot_route	Obligatoire	Spécifie soit la route actuelle au format « SRV_ADDR:SRV_PORT », soit la constante « REFUSED » lorsque les requêtes provenant de l'automate doivent être refusées.	« Host_1:8001 » « REFUSED »
HR.req_time	Obligatoire	Horodatage de la dernière requête traitée.	
HR.noping_req_time	Obligatoire	Horodatage de la dernière requête de type « PING » traitée.	

Tableau 6 — Format de la table des routes courantes (HR).

2.1.5.6 ALGORITHME DE ROUTAGE

2.1.5.6.1 ENRÔLEMENT OU RÉ-ENRÔLEMENT D'UN AUTOMATE

Les premières requêtes émises par un automate sont routées statiquement par la fonction de routage via attribution d'une route par défaut dite « route pilote » entre ledit automate et un serveur (Figure 12 / Pastille E ; Figure 13 / Pastilles E, E1, E2, E3).

L'attribution de cette route pilote s'accompagne de la création ou modification d'une route courante HR associée audit automate (Figure 12 / Pastille E / Flux 9 ; Figure 13 / Pastille E2 / flux 9).

L'automate est dit « **complètement enrôlé »** ou simplement « **enrôlé »,** auprès de la fonction de compilation, s'il possède une route courante HR et si cette dernière définit la version de l'automate (champ HR.atm_version). Dans ce cas, la fonction de compilation voit ledit automate et peut produire des ordres de déroutages le concernant, ce qui permet de router dynamiquement les flux dudit automate.

L'enrôlement est dit « **incomplet »** ou « **partiel »** lorsque la route courante HR existe sans que la version HR.atm_version de l'automate soit définie. Dans ce cas, la fonction de compilation ne voit pas ledit automate tant que son enrôlement n'a pas été complété. Le routage dudit automate reste statique jusqu'à complétude de l'enrôlement.

En d'autres termes, la fonction d'enrôlement et ré-enrôlement d'un automate est un procédé qui consiste via la fonction F_PILOTE à attribuer une route pilote audit automate (Figure 13 / Pastille E1) puis à créer ou modifier la route courante HR associée audit automate (Figure 13 / Pastille E2) dans les cas suivants :

- Lorsque l'automate passe de l'état non enrôlé à l'état partiellement ou complètement enrôlé. En d'autres termes lorsque l'automate effectue sa première requête c.-à-d. lorsqu'il n'existe pas encore de route courante HR associée audit automate.
- Lorsque l'automate passe de l'état partiellement enrôlé à l'état complètement enrôlé. En d'autres termes lorsque l'automate complète son enrôlement c.-à-d. lorsque il est déjà partiellement enrôlé et communique sa version Q.atm_version lors d'une requête Q.
- Lorsque l'automate change d'identité c.-à-d. lorsque l'automate est déjà partiellement ou complétement enrôlé et communique dans une requête Q, soit un identifiant Q.atm_id contredisant l'identifiant mémorisé dans HR.atm_id, soit une version Q.atm_version contredisant la version mémorisée dans HR.atm_version si cette dernière est définie.

Ce procédé d'enrôlement complet et partiel possède l'avantage de router immédiatement un automate sans action du compilateur et sans disposer d'un référentiel des automates optionnel, avec l'inconvénient que le routage reste statique et ne respecte pas les règles (affinités, exclusions, topologie) tant que l'enrôlement n'est pas complet.

Ce procédé d'enrôlement est décrit en détail aux chapitres §2.1.5.6.9 et suivants.

2.1.5.6.2 DÉROUTAGE D'UN AUTOMATE ENRÔLÉ

Ce procédé consiste pour la fonction de routage, à dérouter un automate depuis un serveur vers un autre en réponse à un ordre de déroutage NR produit par la fonction de compilation (Figure 12 / Pastille D ; Figure 13 / Pastilles D et D1 à D6).

Seul peut être dérouté un automate vérifiant simultanément les deux conditions suivantes :

1. L'automate est totalement enrôlé.
2. L'automate ne nécessite pas un ré-enrôlement, c.-à-d. la requête reçue et susceptible d'être déroutée ne caractérise pas un changement d'identité de l'automate.

Si l'ordre de déroutage NR est spécifié immédiat (c.-à-d. si NR.is_immediate est VRAI) en réponse par exemple à un incident, alors la route courante est modifiée immédiatement pour suivre la nouvelle route demandée. Si cette dernière prend la valeur « REFUSED », alors la requête est refusée immédiatement.

Dans le cas contraire (c.-à-d. si l'ordre de déroutage NR est spécifié non urgent c.-à-d. si NR.is_immediate est FAUX) en réponse par exemple à une décision humaine (création d'une règle par un exploitant), alors :

1. Si la requête Q émise par l'automate caractérise soit une fin de session (c.-à-d. Q.is_eos est VRAI) de l'utilisateur soit un dépassement du délai d'attente maximum (timeout) de cette session, alors la route courante est modifiée immédiatement pour suivre la nouvelle route demandée. Si cette dernière prend la valeur « REFUSED », alors la requête est refusée immédiatement.
2. Dans le cas contraire, le routage est maintenu jusqu'à ce que les conditions d'un déroutage soient réunies.

Ce procédé de déroutage est décrit en détail aux chapitres §2.1.5.6.13 et suivants.

2.1.5.6.3 MAINTIEN DE LA ROUTE COURANTE D'UN AUTOMATE PARTIELLEMENT OU COMPLÈTEMENT ENRÔLÉ

Lorsque un automate pour lequel existe une route courante HR, ne satisfait ni les conditions d'un enrôlement ou ré-enrôlement (Figure 13 / Pastille E) ni celles d'un déroutage (Figure 13 / Pastille D) alors ladite requête est traitée par la chaîne de maintien de route (Figure 13 / Pastille M).

Le procédé consiste à maintenir le routage vers la route courante HR.hot_route existante.

Si la route courante HR.hot_route prend la valeur « REFUSED » alors F_ROUTE ne route aucune requête provenant de cet automate et le lui signifie par un code HTTP en réponse indiquant une indisponibilité de type temporaire, ce qui incite le progiciel automate à réessayer ultérieurement.

Ce procédé de maintien est décrit en détail aux chapitres §2.1.5.6.20 et suivants.

2.1.5.6.4 CINÉMATIQUE DÉTAILLÉE DE L'ALGORITHME DE ROUTAGE

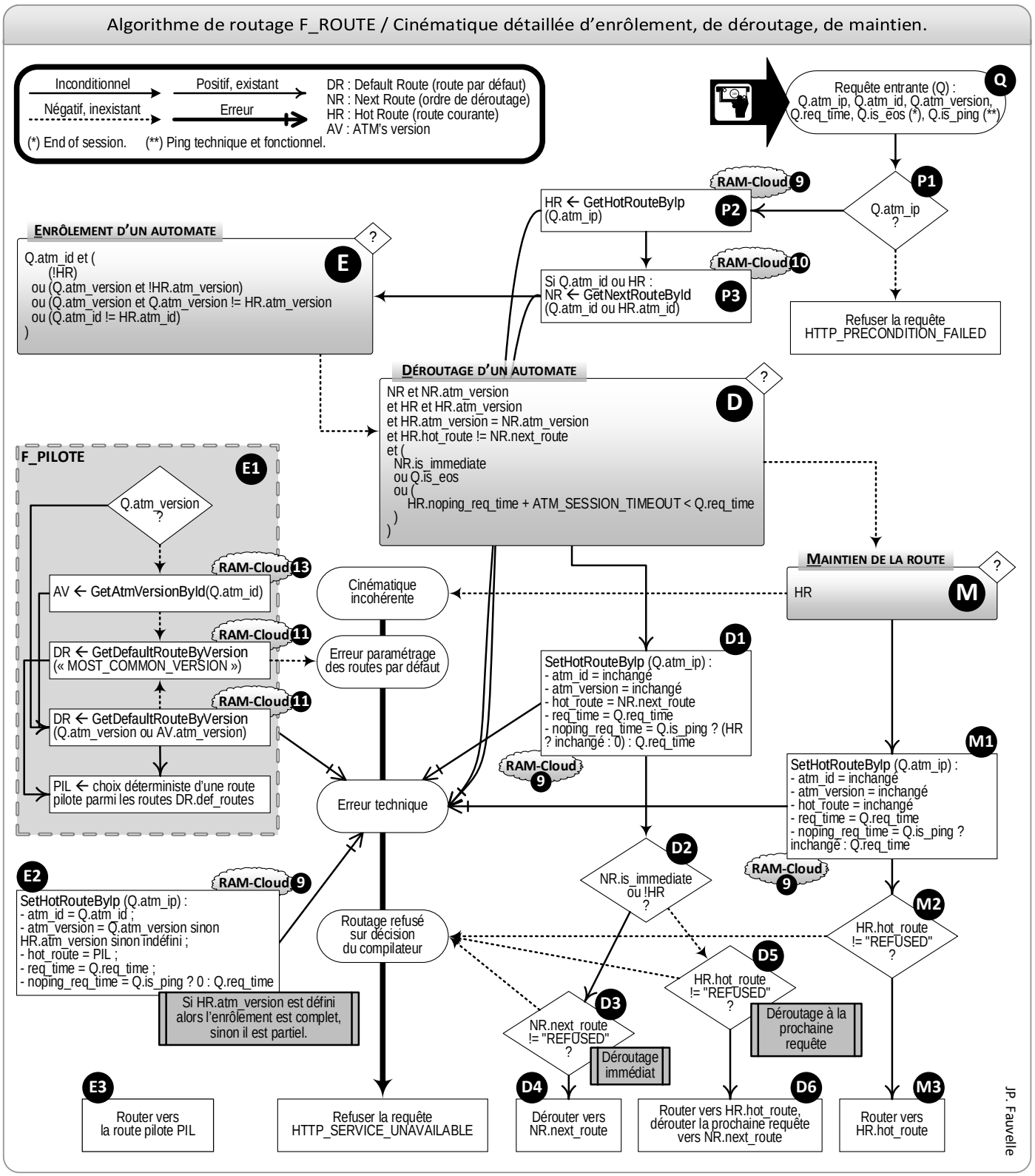

Figure 13 — Algorithme de routage / Cinématique détaillée d'enrôlement, déroutage, maintien d'un automate.

Les différentes étapes de l'algorithme de routage F_ROUTE sont schématisées en Figure 13 ci-avant, et détaillées ci-après.

2.1.5.6.5 ÉTAPE Q : REQUÊTE ENTRANTE ÉMISE PAR L'AUTOMATE

Cette étape de l'algorithme F_ROUTE reçoit une requête Q émise par un automate (Figure 13 / Pastille Q).

La fonction F_ROUTE extrait les informations utiles (Q.atm_ip, Q.atm_id, Q.atm_version, Q.req_time, Q.is_eos, Q.is_ping) dont certaines sont transmises à chaque requête tandis que d'autres sont transmises occasionnellement.

Puis le traitement de la requête Q se poursuit à l'étape P1 (§2.1.5.6.6).

2.1.5.6.6 ÉTAPE P1 : VÉRIFIER L'ADRESSE IP DE L'AUTOMATE

Cette étape de l'algorithme F_ROUTE vérifie l'adresse IP de l'automate (Figure 13 / Pastille P1).

La fonction F_ROUTE vérifie l'existence et la validité (plage d'adresse autorisée, sécurité, etc.) de l'adresse IP de l'automate, en appliquant les opérations ci-après dans cet ordre :

1. Si le champ Q.atm_ip extrait de la requête Q, n'est pas défini, alors F_ROUTE met fin au traitement de la requête Q et retourne une erreur à l'automate.
2. Dans le cas contraire, le traitement se poursuit à l'étape P2 (§2.1.5.6.7).

2.1.5.6.6.1 AMÉLIORATION POSSIBLE

Une amélioration possible consiste à effectuer lors de cette étape, des tests relatifs à la sécurité : format des paramètres de la requête, taille des paramètres, format de l'URI, test de déni de service, et autres opérations de durcissement.

Par exemple, dans le cas du routage des flux d'automates bancaires, cette étape peut implémenter la norme de sécurité PCI-DSS de l'industrie des cartes de paiements.

2.1.5.6.7 ÉTAPE P2 : LIRE LA ROUTE COURANTE ASSOCIÉE À L'AUTOMATE

Cette étape de l'algorithme F_ROUTE lit la route courante associée à l'automate (Figure 13 / Pastille P2).

La fonction F_ROUTE lit dans le RAM-Cloud la route courante, si elle existe, associée à l'adresse IP de l'automate qui est à l'origine de la requête Q, en exécutant les opérations ci-après dans cet ordre :

1. La fonction F_ROUTE lit dans le RAM-Cloud la paire HR notée `{ clé = (atm_ip) ; données = (atm_id, atm_version, hot_route, req_time, noping_req_time) }` dont la clé primaire HR.atm_ip est égale à Q.atm_ip :
    a. Si la lecture est en erreur, alors F_ROUTE met fin au traitement de la requête Q et retourne une erreur à l'automate.
    b. Si la lecture est positive, alors l'objet ainsi lu contient les informations sur la route courante (Tableau 6) : ledit objet est propagé comme tel sous le nom « HR » dans la suite du traitement F_ROUTE qui se poursuit à l'étape P3 (§2.1.5.6.8).
    c. Si la lecture est négative, alors l'objet n'existe pas : ledit objet inexistant est propagé comme tel sous le nom « HR » dans la suite du traitement F_ROUTE qui se poursuit à l'étape P3 (§2.1.5.6.8).

2.1.5.6.8 ÉTAPE P3 : LIRE L'ORDRE DE DÉROUTAGE ASSOCIÉ À L'AUTOMATE

Cette étape de l'algorithme F_ROUTE lit l'ordre de déroutage associé à l'automate (Figure 13 / Pastille P3).

La fonction F_ROUTE lit dans le RAM-Cloud l'ordre de déroutage, s'il existe, associé à l'identifiant de l'automate qui est à l'origine de la requête Q, en exécutant les opérations ci-après dans cet ordre :

1. Si la requête Q contient un champ Q.atm_id ou si HR existe (auquel cas HR contient un champ HR.atm_id défini) alors la fonction F_ROUTE lit dans le RAM-Cloud la paire NR notée `{ clé = (atm_id) ; données = (next_route, is_immediate, atm_version) }` dont la clé

primaire NR.atm_id est égale soit à Q.atm_id si ce champ est défini soit à HR.atm_id dans le cas contraire :

a. Si la lecture est en erreur, alors F_ROUTE met fin au traitement de la requête et retourne une erreur à l'automate.

b. Si la lecture est positive, alors l'objet ainsi lu contient les informations constituant l'ordre de déroutage (Tableau 5) : ledit objet est propagé comme tel sous le nom « NR » dans la suite du traitement F_ROUTE qui se poursuit à l'étape E (§2.1.5.6.9).

c. Si la lecture est négative, alors l'objet n'existe pas : ledit objet inexistant est propagé comme tel sous le nom « NR » dans la suite du traitement F_ROUTE qui se poursuit à l'étape E (§2.1.5.6.9).

2. Dans le cas contraire, alors l'objet n'existe pas et il est propagé comme tel sous le nom « NR » dans la suite du traitement F_ROUTE qui se poursuit à l'étape E (§2.1.5.6.9).

2.1.5.6.9 ÉTAPE E : DÉCIDER SI LES CONDITIONS D'ENRÔLEMENT SONT SATISFAITES

Cette étape de l'algorithme F_ROUTE détermine si l'automate doit être enrôlé ou ré-enrôlé (Figure 13 / Pastille E).

La fonction F_ROUTE détermine si les conditions d'un enrôlement ou d'un ré-enrôlement sont réunies, en exécutant les opérations ci-après dans cet ordre :

1. Si la requête Q ne contient pas un champ défini Q.atm_id alors il n'est pas possible d'enrôler ou ré-enrôler l'automate : le traitement continue à l'étape D (§2.1.5.6.13).
2. Si HR n'existe pas alors l'automate se connecte pour la première fois donc il convient de l'enrôler : le traitement se poursuit à l'étape E1 (§2.1.5.6.10).

   Remarque : lorsqu'un automate change d'adresse IP, sa nouvelle adresse Q.atm_ip n'est la clé primaire d'aucune route courante HR existante. Cette absence de HR correspond au prérequis du point 2 présent, ce qui a pour effet d'enrôler l'automate avec sa nouvelle adresse IP comme s'il s'agissait d'un nouvel automate.
3. Si (Q contient un champ Q.atm_version défini) et si (HR.atm_version est indéfini) alors l'automate est déjà enrôlé partiellement et sa version qui n'était pas connue vient d'être communiquée via le champ Q.atm_version, ce qui permet de compléter l'enrôlement dudit automate : le traitement se poursuit à l'étape E1 (§2.1.5.6.10).
4. Si (Q contient un champ défini Q.atm_version différent de la version déjà connue HR.atm_version, alors l'automate est déjà enrôlé mais il a changé de version progicielle ce qui oblige à le ré-enrôler : le traitement se poursuit à l'étape E1 (§2.1.5.6.10).
5. Si (Q.atm_id est différent de HR.atm_id) alors l'automate est déjà enrôlé mais il a changé d'identifiant ce qui oblige à le ré-enrôler : le traitement se poursuit à l'étape E1 (§2.1.5.6.10).
6. Sinon, dans tous les autres cas, le traitement se poursuit à l'étape D (§2.1.5.6.13).

2.1.5.6.10 ÉTAPE E1 : DÉTERMINER UNE ROUTE PILOTE PAR DÉFAUT LORS D'UN ENRÔLEMENT

Cette étape de la fonction F_ROUTE choisit une route pilote pour l'automate, via une fonction F_PILOTE incluse dans F_ROUTE (Figure 13 / Pastille E1).

En réponse à un automate A émettant une requête Q, la fonction F_PILOTE détermine une route pilote P par défaut ou doublement par défaut (§2.1.5.5.2) en exécutant les opérations ci-après dans cet ordre :

1. Si la requête Q émise par l'automate A contient un champ Q.atm_version défini, alors le traitement se poursuit au point 4 ci-après, sinon au point 2 ci-après.

2. F_PILOTE lit dans le RAM-Cloud (Figure 12 et Figure 13 / Flux 13) la paire AV notée `{ clé = (atm_id) ; donnée = (atm_version) }` dont la clé AV.atm_id est égale à Q.atm_id :
    a. Si la lecture est positive, alors AV.atm_version contient la version du progiciel équipant l'automate A : le traitement se poursuit au point 4 ci-après.
    b. Si la lecture est négative, alors le traitement se poursuit au point 3 ci-après.
    c. Si la lecture est en erreur, alors F_ROUTE met fin au traitement de la requête Q et retourne une erreur à l'automate.
3. F_PILOTE lit dans le RAM-Cloud (Figure 12 et Figure 13 / Flux 11) la paire DR notée `{ clé = (atm_version) ; donnée = (def_routes) }` dont la clé DR.atm_version est égale à « MOST_COMMON_VERSION » :
    a. Si la lecture est positive, alors DR.def_routes contient la liste des routes doublement par défaut probablement compatibles avec le plus grand nombre d'automates : le traitement se poursuit au point 5 ci-après.
    b. Si la lecture est négative, alors il n'existe pas de routes doublement par défaut ce qui constitue une insuffisance de paramétrage : F_ROUTE met fin au traitement de la requête Q et retourne une erreur à l'automate.
    c. Si la lecture est en erreur, alors F_ROUTE met fin au traitement de la requête Q et retourne une erreur à l'automate.
4. F_PILOTE lit dans le RAM-Cloud (Figure 12 et Figure 13 / Flux 11) la paire DR notée `{ clé = (atm_version) ; donnée = (def_routes) }` dont la clé DR.atm_version est égale soit à Q.atm_version s'il est défini soit à AV.atm_version dans le cas contraire – l'un de ces 2 champs étant nécessairement défini (points 1 et 2 ci-avant) :
    a. Si la lecture est positive, alors DR.def_routes contient la liste des routes par défaut assurément compatibles avec l'automate A : le traitement se poursuit au point 5 ci-après.
    b. Si la lecture est négative, alors il n'existe pas de route par défaut associé à l'automate A : le traitement se poursuit au point 3 ci-avant pour tenter d'attribuer une route doublement par défaut.
5. F_PILOTE choisi une des routes dans la liste DR.def_routes calculée au point 3 ou 4 ci-avant. Ce choix est déterministe et uniforme, effectué par hachage sur l'adresse IP (Q.atm_ip) de l'automate. Ce déterminisme est nécessaire afin que plusieurs instances de F_ROUTE traitant plusieurs requêtes provenant d'un même automate, choisissent la même route pilote par défaut.

   Cette route pilote par défaut est propagée comme telle sous le nom « PIL » dans la suite du traitement F_ROUTE qui se poursuit à l'étape E2 (§2.1.5.6.11).

2.1.5.6.11 ÉTAPE E2 : ÉCRIRE LA ROUTE COURANTE LORS D'UN ENRÔLEMENT

Cette étape de l'algorithme F_ROUTE écrit la route courante associée l'automate (Figure 12 / Pastille E / Flux 9 ; Figure 13 / Pastille E2 / Flux 9).

Pour rappel, lors de l'étape P2, la route courante associée à l'adresse IP de l'automate a été lue depuis le RAM-Cloud, puis affectée à un objet nommé HR propagé entre les différentes étapes jusqu'à la présente étape. Depuis, les informations contenues dans les champs de la route courante sont identiques dans l'objet HR et dans le RAM-Cloud.

À présent, la fonction F_ROUTE crée ou met à jour dans le RAM-Cloud, la route courante associée à l'adresse IP (Q.atm_ip) de l'automate, en valorisant ses champs comme ci-après :

Champ	←	Valeur	Condition	Remarque
atm_ip	←	Q.atm_ip	Inconditionnel	Clé primaire.
atm_id	←	Q.atm_id	Inconditionnel	Toujours défini (prérequis de l'étape E).
atm_version	←	Q.atm_version	Q.atm_version est défini	
	←	Inchangé	Q.atm_version est indéfini et HR existe	
	←	indéfini	Q.atm_version est indéfini et HR n'existe pas	
hot_route	←	PIL	Inconditionnel	Toujours défini (résultat de l'étape E1).
req_time	←	Q.req_time	Inconditionnel	Toujours défini.
noping_req_time	←	0	Q.is_ping est VRAI	Toujours défini.
	←	Q.req_time	Q.is_ping est FAUX	Toujours défini.

Tableau 7 — Écriture d'une route courante lors d'un enrôlement.

À ce stade, les informations contenues dans les champs de la route courante ne sont plus identiques dans l'objet HR et dans le RAM-Cloud.

Cette route courante qui vient d'être créée ou mise à jour dans le RAM-Cloud contient la future valeur qui sera affectée à l'objet HR lors de la prochaine requête traitée pour le même automate. Elle sera lue à l'étape P2, donc prise en compte, lors du traitement par F_ROUTE de la prochaine requête émise par cet automate.

Suite à cette écriture :

1. Si l'écriture est en erreur, alors F_ROUTE met fin au traitement de la requête Q et retourne une erreur à l'automate.
2. Dans le cas contraire, alors le traitement se poursuit à l'étape E3 (§2.1.5.6.12).

2.1.5.6.12 ÉTAPE E3 : ROUTER LA REQUÊTE LORS D'UN ENRÔLEMENT

Cette étape de l'algorithme F_ROUTE procède au routage de la requête de l'automate (Figure 13 / Pastille E4).

La fonction F_ROUTE route la requête Q vers la route pilote PIL.

Cette action termine le processus d'enrôlement et le traitement de la requête Q.

2.1.5.6.13 ÉTAPE D : DÉCIDER SI LES CONDITIONS DE DÉROUTAGE SONT SATISFAITES

Cette étape de l'algorithme F_ROUTE détermine si les prérequis d'un déroutage sont remplis (Figure 13 / Pastille D).

Le déroutage d'un automate consiste à modifier sa route courante vers une nouvelle route, cette dernière pouvant prendre la valeur « REFUSED » lorsque la finalité consiste à refuser les requêtes dudit automate.

La fonction F_ROUTE détermine si les conditions d'un déroutage sont réunies, en exécutant les opérations ci-après dans cet ordre :

1. Si (NR existe et NR.atm_version est définie)

   et si (HR existe et HR.atm_version est définie)

   et si (HR.atm_version est égal à NR.atm_version)

   et si (HR.hot_route est différent de NR.next_route)

et si (

NR.is_immediate est VRAI

ou Q.is_eos est VRAI

ou (HR.noping_req_time + P_DUREE_MAX_SESSION_ATM < Q.req_time)

),

alors l'automate doit être dérouté : le traitement se poursuit à l'étape D1 (§2.1.5.6.14).

Le paramètre configurable P_DUREE_MAX_SESSION_ATM correspond à la durée maximum autorisée pour une session automate, au-delà de laquelle la fonction F_ROUTE est autorisée à couper la session de l'utilisateur en déroutant les requêtes.

2. Sinon, dans tous les autres cas, le traitement se poursuit à l'étape M (§2.1.5.6.20).

2.1.5.6.14 ÉTAPE D1 : METTRE À JOUR LA ROUTE COURANTE LORS D'UN DÉROUTAGE

Cette étape de l'algorithme F_ROUTE écrit la route courante associée à l'automate (Figure 13 / Pastille D1).

Pour rappel, lors de l'étape P2, la route courante associée à l'adresse IP de l'automate a été lue depuis le RAM-Cloud, puis affectée à un objet nommé HR propagé entre les différentes étapes jusqu'à la présente étape. Depuis, les informations contenues dans les champs de la route courante sont identiques dans l'objet HR et dans le RAM-Cloud.

À présent, la fonction F_ROUTE met à jour dans le RAM-Cloud, la route courante existante associée à l'adresse IP (Q.atm_ip) de l'automate, en valorisant ses champs comme ci-après :

Champ	←	**Valeur**	**Condition**	**Remarque**
atm_ip	←	Q.atm_ip	Inconditionnel	Clé primaire.
atm_id	←	Inchangé	Inconditionnel	Toujours défini car HR existe (prérequis de l'étape D) et HR.atm_id est un champ obligatoire.
atm_version	←	Inchangé	Inconditionnel	Toujours défini car HR existe (prérequis de l'étape D) et l'automate est enrôlé (prérequis de l'étape D).
hot_route	←	NR.next_route	Inconditionnel	Toujours défini (prérequis de l'étape D). La valeur est une route ou « REFUSED ».
req_time	←	Q.req_time	Inconditionnel	Toujours défini.
noping_req_time	←	Inchangé	Q.is_ping est VRAI et HR existe	Toujours défini.
	←	0	Q.is_ping est VRAI et HR n'existe pas	
	←	Q.req_time	Q.is_ping est FAUX	

Tableau 8 — Écriture d'une route courante lors d'un déroutage.

À ce stade, les informations contenues dans les champs de la route courante ne sont plus identiques dans l'objet HR et dans le RAM-Cloud.

Cette route courante qui vient d'être mise à jour dans le RAM-Cloud contient la future valeur qui sera affectée à l'objet HR lors de la prochaine requête traitée pour le même automate. Elle sera lue à l'étape P2, donc prise en compte, lors du traitement par F_ROUTE de la prochaine requête émise par cet automate.

Suite à cette écriture :

1. Si l'écriture est en erreur, alors F_ROUTE met fin au traitement de la requête Q et retourne une erreur à l'automate.
2. Dans le cas contraire, alors le traitement se poursuit à l'étape D2 (§2.1.5.6.15).

2.1.5.6.15 Étape D2 : gérer le caractère d'immédiateté lors d'un déroutage

Cette étape de l'algorithme F_ROUTE aiguille le traitement de la requête selon le caractère d'immédiateté de l'ordre de déroutage (Figure 13 / Pastille D2).

La fonction F_ROUTE décide si le déroutage peut être exécuté immédiatement ou s'il doit être différé, en exécutant les opérations ci-après dans cet ordre :

1. Si le champ NR.is_immediate est VRAI, alors le déroutage est urgent donc F_ROUTE est autorisée à interrompre une session en cours : le traitement se poursuit à l'étape D3 (§2.1.5.6.16).
2. Dans le cas contraire, alors F_ROUTE n'est pas autorisée à interrompre une session en cours : le traitement se poursuit à l'étape D5 (§2.1.5.6.18).

2.1.5.6.16 Étape D3 : gérer les routes refusées lors d'un déroutage immédiat

Cette étape de l'algorithme F_ROUTE aiguille le traitement de la requête selon que l'ordre de déroutage est une route effective ou une interdiction (Figure 13 / Pastille D3).

La fonction F_ROUTE décide si le déroutage doit être refusé, en exécutant les opérations ci-après dans cet ordre :

1. Si le champ NR.next_route a pour valeur « REFUSED », alors le compilateur proscrit le routage des requêtes provenant de cet automate : F_ROUTE met fin au traitement de la requête Q et retourne une erreur à l'automate.

   La présente étape correspond à un déroutage immédiat, raison pour laquelle la valeur « REFUSED » est testée sur la route cible (NR) et non sur la route courante (HR).
2. Dans le cas contraire, alors le déroutage immédiat peut être exécuté : le traitement se poursuit à l'étape D4 (§2.1.5.6.17).

2.1.5.6.17 Étape D4 : dérouter la requête immédiatement

Cette étape de l'algorithme F_ROUTE exécute le routage de l'automate (Figure 13 / Pastille D4).

La fonction F_ROUTE déroute la requête Q vers la nouvelle route NR.next_route ce qui a pour effet de rendre le déroutage immédiatement effectif.

Cette action termine le déroutage immédiat et le traitement de la requête Q.

La nouvelle route a déjà été affectée à la route courante dans le RAM-Cloud lors de l'étape D1, ce qui l'a rendue permanente : les prochaines requêtes provenant de cet automate continueront à suivre cette nouvelle route.

2.1.5.6.18 Étape D5 : gérer les routes refusées lors d'un déroutage différé

Cette étape de l'algorithme F_ROUTE aiguille le traitement de la requête selon que l'ordre de déroutage est une route effective ou une interdiction (Figure 13 / Pastille D5).

La fonction F_ROUTE décide si le déroutage doit être refusé, en exécutant les opérations ci-après dans cet ordre :

1. Si le champ HR.hot_route a pour valeur « REFUSED », alors le compilateur proscrit le routage des requêtes provenant de cet automate : F_ROUTE met fin au traitement de la requête Q et retourne une erreur à l'automate.

   La présente étape correspond à un déroutage différé, raison pour laquelle la valeur « REFUSED » est testée sur la route courante (HR) et non sur la route cible (NR).

2. Dans le cas contraire, alors le déroutage différé peut être exécuté : le traitement se poursuit à l'étape D6 (§2.1.5.6.19).

2.1.5.6.19 ÉTAPE D6 : DÉROUTER LA REQUÊTE EN DIFFÉRÉ

Cette étape de l'algorithme F_ROUTE exécute le routage de l'automate (Figure 13 / Pastille D6).

La fonction F_ROUTE route la requête Q vers la route courante HR.hot_route ce qui a pour effet de maintenir la route existante.

Cette étape termine le déroutage différé et le traitement de la requête Q.

Cette requête est la dernière qui suit la route courante pour cet automate : la prochaine requête suivra la nouvelle route demandée.

En effet, la nouvelle route NR.next_route a déjà été affectée à la route courante dans le RAM-Cloud lors de l'étape D1, ce qui l'a rendue permanente. Le déroutage sera donc effectif dès la prochaine requête de cet automate, au moment de la lecture de la route courante lors de l'étape P2.

2.1.5.6.20 ÉTAPE M : DÉCIDER SI LES CONDITIONS DE MAINTIEN SONT SATISFAITES

Cette étape de l'algorithme F_ROUTE détermine si les prérequis d'un maintien de route sont remplis (Figure 13 / Pastille M).

À cette fin, la fonction F_ROUTE exécute les opérations ci-après dans cet ordre :

1. Si HR existe, alors cette route courante HR peut être maintenue : le traitement se poursuit à l'étape M1 (§2.1.5.6.21).
2. Dans le cas contraire, alors F_ROUTE met fin au traitement de la requête Q et retourne une erreur à l'automate.

2.1.5.6.21 ÉTAPE M1 : ÉCRIRE LA ROUTE COURANTE LORS D'UN MAINTIEN DE ROUTE

Cette étape de l'algorithme F_ROUTE écrit la route courante associée à l'automate (Figure 13 / Pastille M1).

Pour rappel, lors de l'étape P2, la route courante associée à l'adresse IP de l'automate a été lue depuis le RAM-Cloud, puis affectée à un objet nommé HR propagé entre les différentes étapes jusqu'à la présente étape. Depuis, les informations contenues dans les champs de la route courante sont identiques dans l'objet HR et dans le RAM-Cloud.

À présent, la fonction F_ROUTE met à jour dans le RAM-Cloud, la route courante existante associée à l'adresse IP (Q.atm_ip) de l'automate, en valorisant ses champs comme ci-après :

Champ	←	Valeur	Condition	Remarque
atm_ip	←	Q.atm_ip	Inconditionnel	Clé primaire.
atm_id	←	Inchangé	Inconditionnel	Toujours défini car HR existe (prérequis de l'étape M) et HR.atm_id est un champ obligatoire.
atm_version	←	Inchangé	Inconditionnel	Défini si l'automate est enrôlé complètement, indéfini si l'automate est enrôlé partiellement.

hot_route	←	Inchangé	Inconditionnel	Route actuelle (y compris « REFUSED ») maintenue. Toujours défini.
req_time	←	Q.req_time	Inconditionnel	Toujours défini.
noping_req_time	←	Inchangé	Q.is_ping est VRAI	Toujours défini.
	←	Q.req_time	Q.is_ping est FAUX	

Tableau 9 — Écriture d'une route courante lors d'un maintien de route.

À ce stade, les informations contenues dans les champs de la route courante ne sont plus identiques dans l'objet HR et dans le RAM-Cloud.

Cette route courante qui vient d'être mise à jour dans le RAM-Cloud contient la future valeur qui sera affectée à l'objet HR lors de la prochaine requête traitée pour le même automate. Ladite valeur sera lue à l'étape P2, donc prise en compte, lors du traitement par F_ROUTE de la prochaine requête émise par cet automate.

Suite à cette écriture :

1. Si l'écriture est en erreur, alors F_ROUTE met fin au traitement de la requête Q et retourne une erreur à l'automate.
2. Dans le cas contraire, alors le traitement se poursuit à l'étape M2 (§2.1.5.6.22).

2.1.5.6.22 ÉTAPE M2 : GÉRER LES ROUTES REFUSÉES LORS D'UN MAINTIEN DE ROUTE

Cette étape de l'algorithme F_ROUTE aiguille le traitement de la requête selon que l'ordre de déroutage est une route ou une interdiction (Figure 13 / Pastille M2).

À cette fin, la fonction F_ROUTE exécute les opérations ci-après dans cet ordre :

1. Si le champ HR.hot_route a pour valeur « REFUSED », alors le compilateur proscrit le routage des requêtes provenant de cet automate : F_ROUTE met fin au traitement de la requête Q et retourne une erreur à l'automate.
2. Dans le cas contraire, alors le déroutage peut être exécuté : le traitement se poursuit à l'étape M3 (§2.1.5.6.23).

2.1.5.6.23 ÉTAPE M3 : ROUTER LA REQUÊTE LORS D'UN MAINTIEN DE ROUTE

Cette étape de l'algorithme F_ROUTE exécute le routage de l'automate (Figure 13 / Pastille M3).

La fonction F_ROUTE route la requête Q vers la route courante HR.hot_route ce qui a pour effet de maintenir la route existante.

Cette étape termine le déroutage de maintien et le traitement de la requête Q.

2.1.5.7 RÉSILIENCE DE LA FONCTION DE ROUTAGE

En cas d'arrêt de la couche Middle, les impacts sur la fonction de routage sont les suivants :

Fonctionnalité	Impact
Maintien des routes	La fonction F_ROUTE continue, sans aucune interruption, à exécuter les routages en cours c.-à-d. le maintien des routes courantes existantes. Les éventuelles requêtes routées vers un serveur qui serait devenu indisponible postérieurement à l'arrêt de la couche Middle, sont maintenues vers ledit serveur, ce qui provoque une indisponibilité des automates dont les

	requêtes restent routées vers ledit serveur indisponible.
Déroutages	La fonction F_ROUTE continue, sans aucune interruption, à exécuter les ordres de déroutages transmis par F_COMPIL avant son indisponibilité. Une fois tous les ordres exécutés, les routages réalisés par F_ROUTE sont des maintiens de routes (point précédent). Le routage reste alors statique jusqu'au rétablissement de la couche Middle et en particulier de la fonction F_COMPIL.
Enrôlements	La fonction F_ROUTE continue sans interruption à enrôler (ou ré-enrôler) les nouveaux automates, en utilisant les tables DR (routes par défaut) et AV (versions des automates) transmises par F_COMPIL avant son indisponibilité. Il est donc possible que l'enrôlement affecte à un automate, une route pilote par défaut vers un serveur qui était disponible avant l'arrêt de la couche Middle mais qui ne l'est plus, provoquant l'indisponibilité dudit automate. Ce mécanisme a l'avantage de maintenir un lissage de charge minimum (statique) sur la base des informations transmises par F_COMPIL avant son arrêt.
Régulation de charge	La régulation dynamique de la charge ne fonctionne plus puisque elle est assurée par F_COMPIL via la publication régulière des ordres de déroutages.
Exploitation	Les éventuelles modifications apportées par l'exploitant aux règles (affinités, exclusion, topologie) et aux référentiels (compatibilité, routes par défaut) via les interfaces IHM de la couche Back ne sont plus prises en compte puisque ces modifications sont acheminées via la couche Middle supposée arrêtée.

Tableau 10 — Résilience de la fonction de routage.

Il est intéressant de souligner que l'arrêt de la couche Middle est sans conséquence dès lors que l'état de disponibilité des serveurs n'évolue pas pendant ledit arrêt.

2.1.5.8 ARCHITECTURE TECHNIQUE

La couche Front est composée d'un nombre quelconque de nœuds informatiques sur lesquels sont distribués les traitements et possède une capacité de mise à l'échelle (*scalability*) horizontale pour les traitements et les données (Figure 14).

La fonction de routage F_ROUTE est également multi instanciée au sein de chaque nœud.

Les données sont stockées dans un moteur NoSQL RAM-Cloud à faible latence, qui distribue ces données sur tous les nœuds Front.

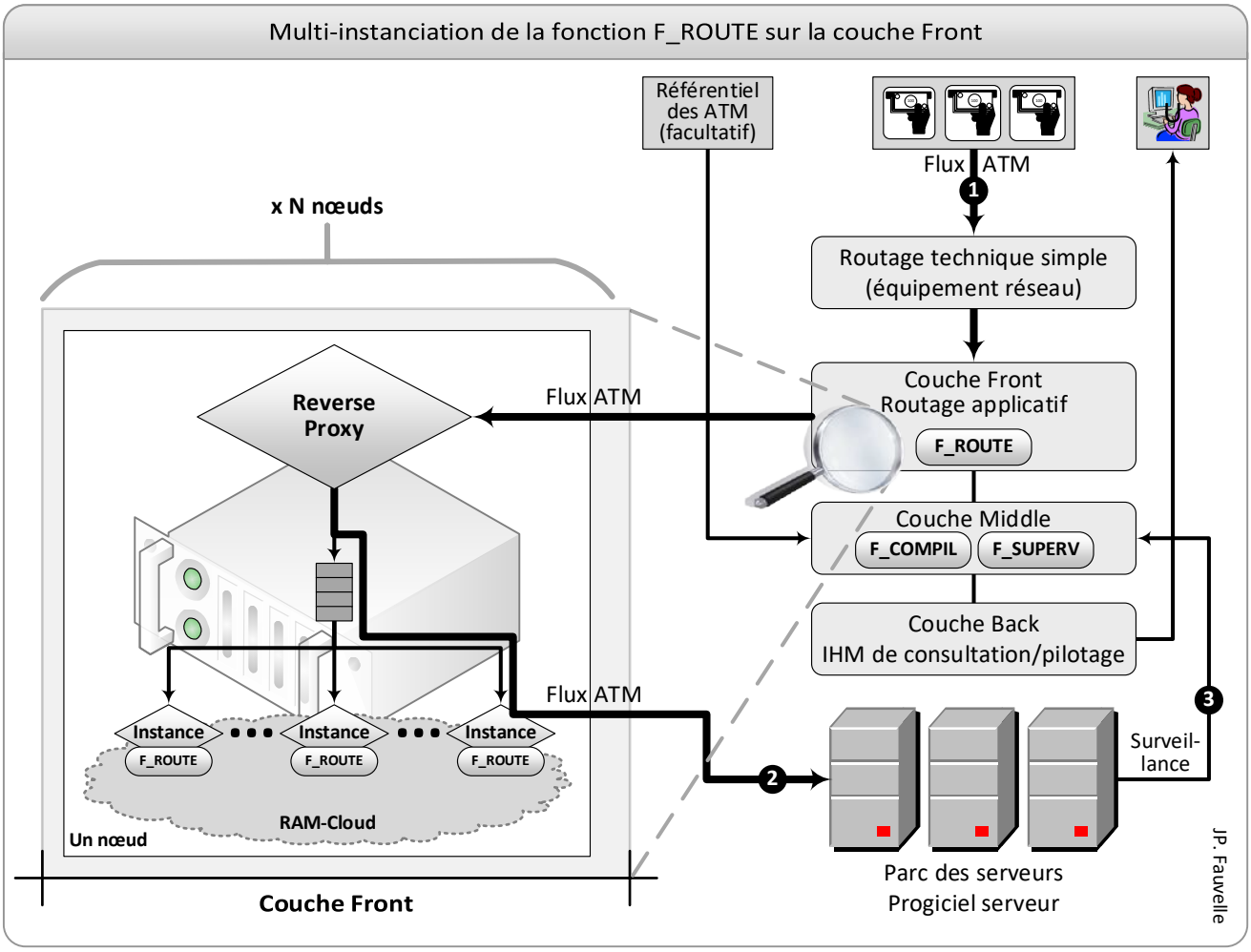

Figure 14 — Multi-instanciation sur la couche Front.

2.1.6 COUCHE « MIDDLE » (F_COMPIL, F_SUPERV)

Cette couche implémente deux fonctions principales (Figure 11 ; Figure 14 ; Figure 17) et stocke ses données dans une base de données relationnelle :

1. La fonction de compilation F_COMPIL, qui produit régulièrement la table DR (routes par défaut), la table AV (versions des automates) et la table NR (ordres de déroutage) à l'usage de la fonction F_ROUTE sur la couche Front.
2. La fonction de surveillance F_SUPERV, qui vérifie régulièrement l'état des serveurs déclarés dans le référentiel des serveurs et enregistre cette information dans sa base de données. Cette information est utilisée par la fonction F_COMPIL pour déterminer quelles routes sont disponibles (serveurs ni incidentés ni suspendus volontairement) afin que la fonction F_COMPIL puisse y faire référence dans les routes par défaut et ordres de déroutages qu'elle produit.

2.1.6.1 PROCÉDÉ DE COMPILATION DES ROUTES EN 6 PHASES

La fonction F_COMPIL implémente l'intelligence du routage applicatif (Figure 15).

Cette fonction est exécutée à fréquence régulière afin de pouvoir gérer l'équilibrage de charge.

Afin de garantir une bonne réactivité, F_COMPIL est également exécutée à chaque évènement pouvant avoir une incidence sur le résultat de la compilation. Ces évènements sont par exemple le changement de disponibilité d'un serveur, l'ajout ou la modification d'une règle, la modification d'un référentiel, et d'une manière générale toute modification des données qui sont utilisées par le processus de compilation.

À chaque cycle de compilation, F_COMPIL exécute tour à tour les différentes phases de compilation dont chacune **propose** des ordres de déroutage concernant tout ou partie de la flotte d'automates. Ainsi, un cycle de compilation peut produire des déroutages pour la totalité de la flotte, ou pour une partie seulement, ou pour aucun automate.

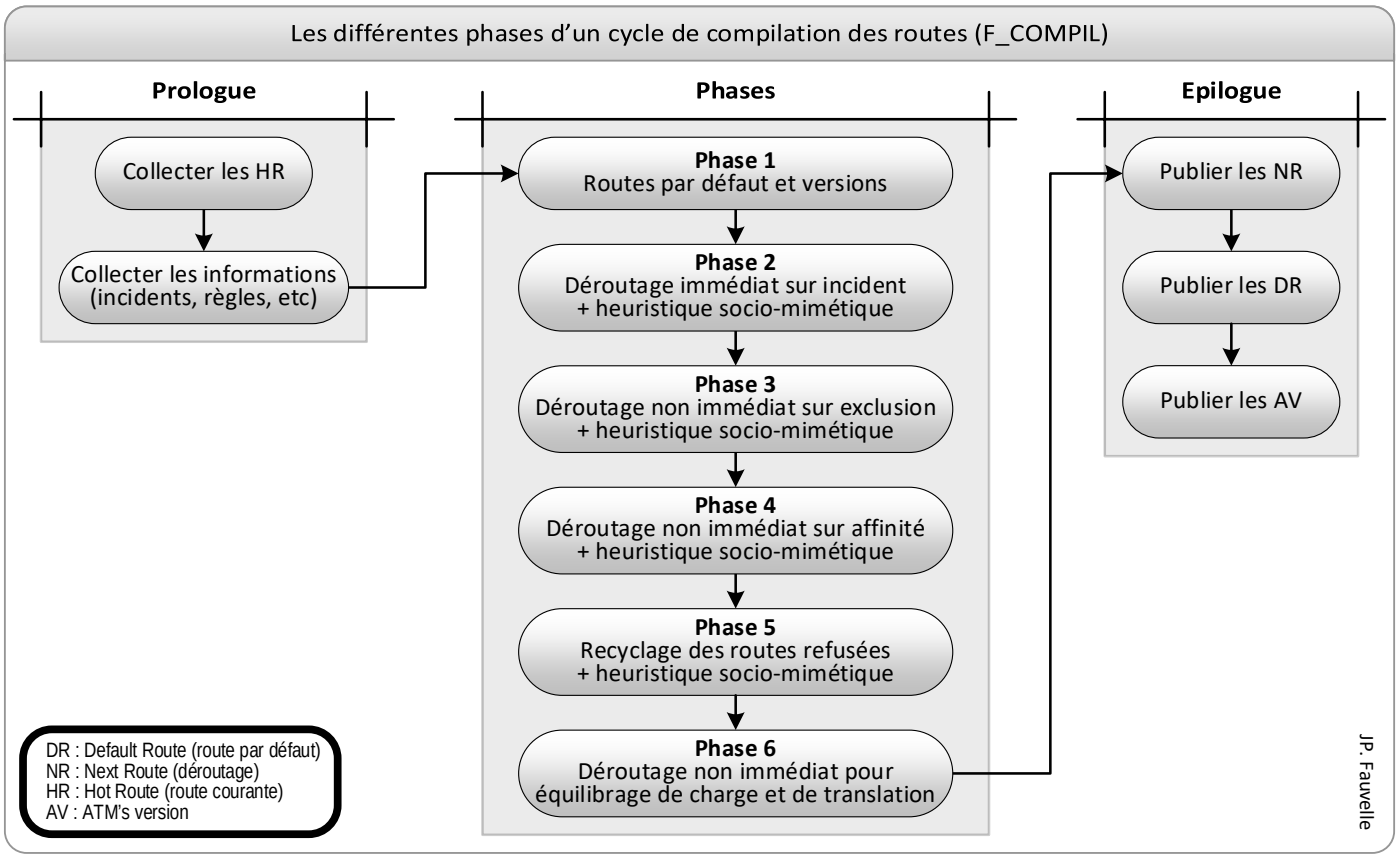

Figure 15 — Les différentes phases d'un cycle de compilation.

Les phases sont numérotées par priorité décroissante sur l'urgence des ordres. Ainsi, lorsqu'une phase du processus de compilation propose un ordre de déroutage concernant un automate donné, alors les phases suivantes du même cycle de compilation ne peuvent plus proposer de déroutage pour ledit automate.

Par exemple, si un ordre de déroutage a été proposé en phase 2 concernant un automate en réponse à un incident sur un serveur, alors la phase 3 ne peut pas proposer un déroutage pour ledit automate pendant ce même cycle de compilation.

Tant que ces ordres **proposés** ne sont pas **publiés** en phase d'épilogue, ils ne sont pas pris en compte par la fonction de routage F_ROUTE.

À l'issue du cycle de compilation, les ordres de déroutage (NR), les routes par défaut (DR) et les versions automates (AV) qui ont été **proposés** au cours des phases de compilation, sont finalement **publiés** de façon atomique vers la couche Front (Figure 12 / Cadre C / Flux 7, 8, 12).

2.1.6.1.1 ASSERVISSEMENT DE LA CHARGE ET DE LA TRANSLATION DE CHARGE

Le procédé ci-après s'applique à un parc de serveurs homogène ou hétérogène en capacité de traitement.

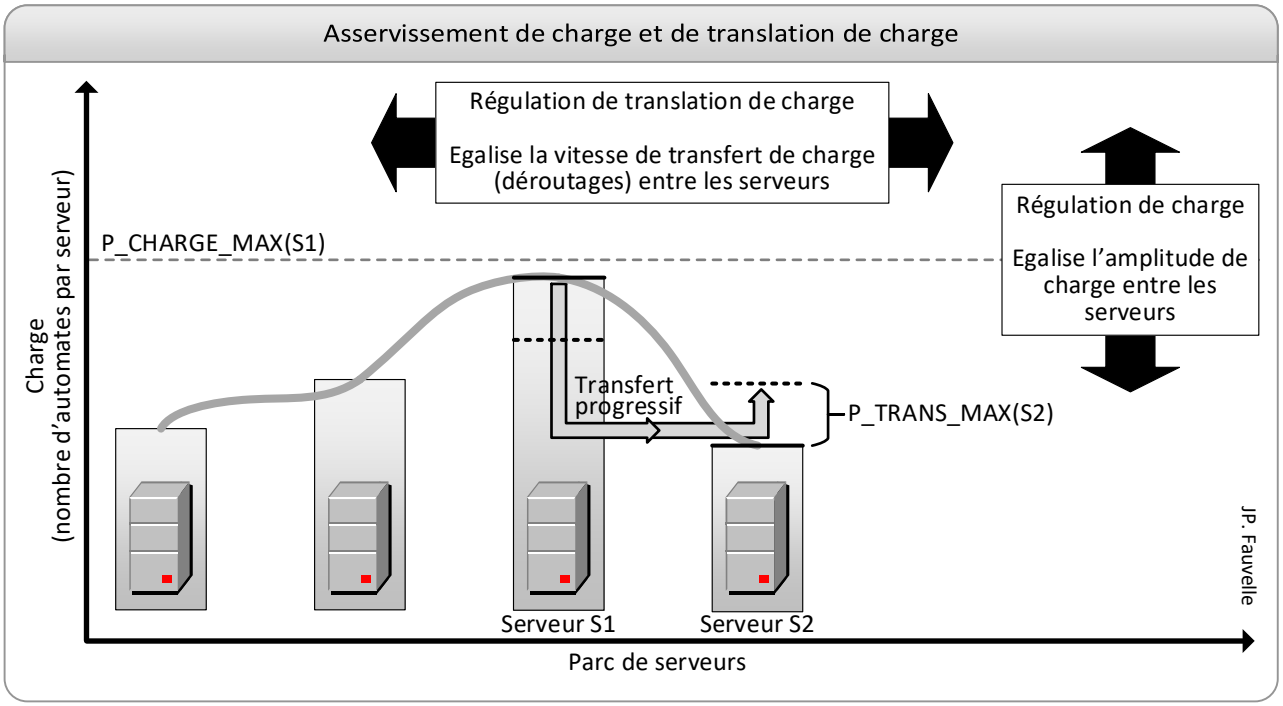

Figure 16 — Asservissement de la charge et de la translation de charge.

La distribution des requêtes sur la plateforme se fait en affectant des automates à des serveurs, via des ordres de déroutage calculés par la fonction F_COMPIL dont l'algorithme respecte deux grands principes :

1. Plafonner le nombre maximum d'automates affectés à chaque serveur afin de lisser et réguler la **distribution de charge** sur le parc de serveurs (Tableau 11 ; Figure 16).
2. Plafonner la valeur maximum de la **balance des déroutages** (entrants / sortants) sur chaque serveur afin de lisser et réguler la **translation de charge** sur le parc de serveurs (Tableau 11 ; Figure 16).

   Ladite balance des déroutages repose sur les concepts suivants :

   a. Un **déroutage entrant** vers un serveur S concernant un automate A, est un ordre de déroutage en attente d'exécution et dont l'exécution à venir aura pour effet de basculer vers S les requêtes provenant de A, ces-dernières étant actuellement soit routées vers un serveur autre que S soit refusées.

   b. Un **déroutage sortant** d'un serveur S concernant un automate A, est un ordre de déroutage en attente d'exécution et dont l'exécution à venir aura pour effet de ne plus router vers S les requêtes provenant de A, parce que lesdites requêtes seront soit basculées vers un serveur autre que S soit refusées.

   c. La **balance des déroutages** pour un serveur donné est le nombre de déroutages entrant vers ledit serveur diminué du nombre de déroutages sortant dudit serveur.

Ce procédé d'asservissement de la balance des déroutages permet par exemple, lorsque une partie importante du parc de serveurs devient subitement indisponible, d'éviter de dérouter subitement tous les automates affectés par l'évènement vers le reste du parc au risque de l'incidenter à son tour par contagion (effet « mouvement de foule »).

Les principaux paramètres et variables d'état nécessaires à l'asservissement de la charge et de la translation de charge sont détaillés ci-après :

Objet	Description
CHARGE(S)	Variable d'état, contenant la valeur instantanée de la charge d'un serveur S, calculée comme le nombre de routes courantes dirigées actuellement vers S. Valeurs autorisées : entier naturel positif ou nul.

<table>
<tr><td></td><td>$\begin{cases} CHARGE \in \mathbb{N} \\ CHARGE \in [0, +\infty[\end{cases}$</td></tr>
<tr><td>P_CHARGE_MAX(S)</td><td>Paramètre configurable, fixant le nombre maximum d'automates que le serveur S est autorisé à servir simultanément à tout instant. En d'autres termes, il s'agit du nombre maximum de routes courantes autorisées vers S.
Valeurs autorisées : entier naturel strictement positif.
$\begin{cases} P\_CHARGE\_MAX \in \mathbb{N} \\ P\_CHARGE\_MAX \in [1, +\infty[\end{cases}$</td></tr>
<tr><td>CHARGE_REL(S)</td><td>Variable d'état, contenant la valeur instantanée de la charge relative d'un serveur S.
Valeurs autorisées : nombre réel compris entre zéro et un, bornes incluses. Toute valeur supérieure à 1 est ramenée à 1.
$CHARGE\_REL(S) = \frac{CHARGE(S)}{P\_CHARGE\_MAX(S)}$ $\begin{cases} CHARGE\_REL \in \mathbb{R} \\ CHARGE\_REL \in [0,1] \end{cases}$</td></tr>
<tr><td>TRANS(S)</td><td>Variable d'état, contenant la valeur instantanée de la balance des déroutages sur un serveur S, calculée comme le nombre de déroutages entrant vers S diminué du nombre de déroutages sortant de S.
Valeurs autorisées : entier naturel. La valeur est positive si l'entrant l'emporte sur le sortant, négative dans le cas contraire, ou nulle en cas d'équilibre.
$\begin{cases} TRANS \in \mathbb{N} \\ TRANS \in]-\infty, +\infty[\end{cases}$</td></tr>
<tr><td>P_TRANS_MAX(S)</td><td>Paramètre configurable, fixant la valeur maximum autorisée pour la balance des déroutages sur un serveur S.
Valeurs autorisées : entier naturel strictement positif.
$\begin{cases} P\_TRANS\_MAX \in \mathbb{N} \\ P\_TRANS\_MAX \in [1, +\infty[\end{cases}$</td></tr>
<tr><td>TRANS_REL(S)</td><td>Variable d'état, contenant la valeur instantanée de la balance relative des déroutages sur un serveur S.
Valeurs autorisées : nombre réel compris entre zéro et un, bornes incluses. Toute valeur négative est ramenée à zéro, toute valeur supérieure à 1 est ramenée à 1.
$TRANS\_REL(S) = \frac{TRANS(S)}{P\_TRANS\_MAX(S)}$ $\begin{cases} TRANS\_REL \in \mathbb{R} \\ TRANS\_REL \in [0,1] \end{cases}$
<table>
<tr><th colspan="4">Exemple</th></tr>
<tr><th colspan="2">Routes courantes</th><th>Déroutages en attente d'exécution</th><th rowspan="2">Remarque</th></tr>
<tr><th>Automate</th><th>Routé vers</th><th>Nouvelle route à venir</th></tr>
<tr><td>A1</td><td>S1</td><td>S2</td><td>Route sortante de S1</td></tr>
<tr><td>A2</td><td>S1</td><td>Pas de déroutage</td><td>Route maintenue vers S1</td></tr>
<tr><td>A3</td><td>S1</td><td>Pas de déroutage</td><td>Route maintenue vers S1</td></tr>
<tr><td>A4</td><td>S1</td><td>Pas de déroutage</td><td>Route maintenue vers S1</td></tr>
<tr><td>A5</td><td>S2</td><td>S1</td><td>Route entrante vers S1</td></tr>
<tr><td>A6</td><td>S3</td><td>S1</td><td>Route entrante vers S1</td></tr>
<tr><td>A7</td><td>« REFUSED »</td><td>S1</td><td>Route entrante vers S1</td></tr>
<tr><td colspan="4">$\mathbf{TRANS\_REL(S1)} = \frac{\boldsymbol{3\ routes\ entrantes\ vers\ S1 - 1\ route\ sortante\ de\ S1}}{\boldsymbol{P\_TRANS\_MAX(S1)}} = \frac{\mathbf{2}}{\mathbf{10}}(*)$</td></tr>
</table>
(*) En fixant par exemple P_TRANS_MAX(S1) à 10.</td></tr>
</table>

P_TRANS_SURCOUT	Paramètre configurable, fixant le surcoût en ressources informatiques consommées sur les serveurs par le traitement des primo-requêtes, par rapport aux non primo-requêtes. Valeurs autorisées : nombre réel strictement positif. $\begin{cases} P\_TRANS\_SURCOUT \in \mathbb{R} \\ P\_TRANS\_SURCOUT \in]0, +\infty[\end{cases}$ À titre d'exemples : 1. Si les primo-requêtes consomment en moyenne **autant** de ressources que les autres requêtes, alors P_TRANS_SURCOUT doit être fixé à 1. 2. Si les primo-requêtes consomment en moyenne 2.5 fois **plus** de ressources que les autres requêtes, alors P_TRANS_SURCOUT doit être fixé à 2.5. 3. Si les primo-requêtes consomment en moyenne 3 fois **moins** de ressources que les autres requêtes, alors P_TRANS_SURCOUT doit être fixé à 1/3 = 0.333.
SATUR(S)	Variable d'état, contenant la valeur instantanée de la saturation globale d'un serveur S, calculée comme la moyenne pondérée, entre d'une part la charge du serveur S affecté d'un poids unitaire, et d'autre part la balance des déroutages du même serveur affectée d'un poids paramétrable P_TRANS_SURCOUT. Valeurs autorisées : nombre réel positif ou nul. $SATUR(S) = \frac{1 * CHARGE(S) + P\_TRANS\_SURCOUT * TRANS(S)}{1 + P\_TRANS\_SURCOUT}$ $\begin{cases} SATUR \in \mathbb{R} \\ SATUR \in [0, +\infty[\end{cases}$ Lors du calcul ci-dessus, toute valeur négative produite par TRANS(S) est ramenée à zéro.

Tableau 11 — Principaux paramètres et variables inhérents à l'asservissement de la charge.

2.1.6.1.1.1 AMÉLIORATION POSSIBLE

Les paramètres définis dans le Tableau 11 caractérisent le fonctionnement et les capacités du progiciel client / serveur, en moyenne.

Or les différentes versions du progiciel qui cohabitent en production pourraient potentiellement présenter une hétérogénéité significative en termes de fonctionnement et de capacité.

Une amélioration consisterait à rendre lesdits paramètres dépendant de la version V du progiciel serveur :

1. P_CHARGE_MAX(S,V) en remplacement de P_CHARGE_MAX(S).
2. P_TRANS_MAX(S,V) en remplacement de P_ TRANS_MAX(S).
3. P_TRANS_SURCOUT(V) en remplacement de P_TRANS_SURCOUT.

2.1.6.1.2 HEURISTIQUE SOCIO-MIMÉTIQUE

Les critères intervenant dans le calcul des déroutages par la fonction F_COMPIL sont nombreux (compatibilité des versions déployées du progiciel, disponibilité des serveurs, règles d'affinités, règles d'exclusions, règles topologiques, régulation de la charge et de la translation de charge).

La fonction F_COMPIL doit également prendre en compte des rétroactions temporelles entre des données (HR, DR, NR, AV) pouvant exister en plusieurs occurrences (en cours d'exécution, générées au cycle précédent de compilation, en cours de calcul pour le cycle courant).

Un algorithme F_COMPIL simpliste pourrait mener certains automates à se retrouver sans possibilité de déroutage alors qu'il reste pourtant de la capacité de traitement sur la plateforme, comme illustré ci-après :

Exemple
Hypothèses de l'exemple : • L'automate A1 est routable vers les serveurs S1 / S2 / S3 / S4 / S5.

- A1 est routé actuellement vers S1.
- Une règle d'exclusion interdit de router A1 vers S3.
- Une règle topologique interdit (condition C_JAMAIS) le déroutage de tout automate depuis S1 vers S4.
- S5 est indisponible.

Il en découle que A1 n'est plus routable que vers S1 / S2 : son seul déroutage possible est donc vers S2.

Si S2 est saturé alors A1 n'est plus déroutable, et ce, même si d'autres serveurs du parc ne sont pas saturés.

Si certains automates occupent de la capacité de traitement sur S2, et sont compatibles avec d'autres serveurs, il est judicieux de les affecter aux autres serveurs non saturés afin de céder leur place à A1 sur S2.

Pour cette raison, les automates qui possèdent à un instant donné le moins de possibilités de déroutages, doivent être déroutés en priorité, plus précisément avec une priorité inversement corrélée au nombre de déroutages possibles.

Ce procédé consistant à servir en priorité les automates les moins « employables » à un instant donné, permet de réduire le nombre d'automates « au chômage technique » en compétition avec les ressources « serveurs ».

Cette **heuristique mimant un comportement social** (« socio-mimétique ») agit dans l'intérêt du plus grand nombre, et permet d'optimiser le nombre d'automates effectivement routés, donc la disponibilité globale moyenne de la flotte d'automates, tout en conservant une durée de compilation raisonnable ne dépassant pas quelques secondes même pour traiter les routes de plusieurs dizaines de milliers d'automates.

Cette méthode ne détermine pas la meilleure solution mathématique mais elle procure une solution efficace de type heuristique. La complexité algorithmique de la fonction F_COMPIL est présentée au §2.1.6.2.

2.1.6.1.3 COMPILATION : PROLOGUE

2.1.6.1.3.1 COLLECTE DES INFORMATIONS REQUISES

Le processus de compilation débute par la collecte des informations listées dans le Tableau 12 ci-après.

Les flux associés sont représentés en Figure 17 et détaillés dans le tableau ci-après :

Flux	Intitulé	Description
IN1	Routes courantes	Ces informations sont collectées depuis le Front (Figure 17 / Flux IN1). Une route courante fait référence à un serveur, ou prend la valeur « REFUSED ». L'ensemble des routes courantes constitue l'exhaustivité des automates enrôlés partiellement ou complétement.
IN2	Ordres de déroutages du précédent cycle de compilation	Ces informations sont collectées sur la base de données de la couche Middle (Figure 17 / Flux IN2 / COPIE_NR). Un ordre de déroutage fait référence à un serveur, ou prend la valeur « REFUSED ». Ces ordres de déroutages ont été publiés vers la couche Front lors du précédent cycle de compilation. La comparaison entre ces ordres publiés précédemment (flux IN2) et les routes courantes (flux IN1) permet de distinguer les ordres déjà exécutés de ceux toujours en attente d'exécution.
IN3	État de disponibilité des serveurs	L'état de disponibilité des serveurs existant est collecté depuis la base de données de la couche Middle (Figure 17 / Flux IN3). Un serveur est disponible s'il n'est ni incidenté ni volontairement suspendu.
IN4	Référentiel de compatibilité	Cette information est collectée depuis la base de données de la couche Middle (Figure 17 / Flux IN4).

		L'information peut être représentée sous la forme d'une matrice à deux dimensions (version du progiciel client, version du progiciel serveur) où chaque valeur dans la matrice est un booléen indiquant la compatibilité entre lesdites versions.
IN5	Règles d'affinités	Cette information est collectée depuis la base de données de la couche Middle (Figure 17 / Flux IN5). Seules les règles actives sont prises en compte. L'information correspondant à l'ensemble des règles actives peut être représentée par une matrice à deux dimensions (identifiant automate, identifiant serveur) où chaque valeur dans la matrice est un booléen indiquant si ledit automate est en affinité avec ledit serveur.
IN6	Règles d'exclusion	Cette information est collectée depuis la base de données de la couche Middle (Figure 17 / Flux IN6). Seules les règles actives sont prises en compte. L'information correspondant à l'ensemble des règles actives peut être représentée par une matrice à deux dimensions (identifiant automate, identifiant serveur) où chaque valeur dans la matrice est un booléen indiquant si ledit automate est interdit de routage vers ledit serveur.
IN7	Règles topologiques	Cette information est collectée depuis la base de données de la couche Middle (Figure 17 / Flux IN7). L'information peut être représentée par une matrice à deux dimensions (identifiant serveur source, identifiant serveur cible) où chaque valeur dans la matrice spécifie sous quelle condition (C_JAMAIS, C_INCIDENT, C_TOUJOURS) tout automate qui serait routé vers ledit serveur source serait autorisé à être dérouté vers ledit serveur cible.
IN8	Référentiel des serveurs	Cette information est collectée depuis la base de données de la couche Middle (Figure 17 / Flux IN8).
IN9	Référentiel des routes par défaut	Cette information est collectée depuis la base de données de la couche Middle (Figure 17 / Flux IN9).
IN10	Référentiel des automates	Cette information est collectée depuis le référentiel des automates de l'entreprise, optionnel. Les éventuelles différences entre ledit référentiel et la flotte réelle des automates en service ont les conséquences suivantes : 1. Si un automate est en service mais non présent dans le référentiel, alors la fonction F_PILOTE est moins efficace lors du calcul d'une route pilote par défaut, mais sans conséquence si la route doublement par défaut est correctement paramétrée. 2. Si un automate est présent dans le référentiel mais non en service, cela n'a aucune conséquence. 3. Si un automate est en service et présent dans le référentiel avec une version incorrecte, alors la fonction F_PILOTE pourrait attribuer une route pilote par défaut erronée, mais seulement de façon temporaire puisque l'automate communique lui-même régulièrement sa version dans ses requêtes et que cette information est prioritaire sur celle provenant du référentiel.

Tableau 12 — Données collectées lors du prologue débutant un cycle de compilation.

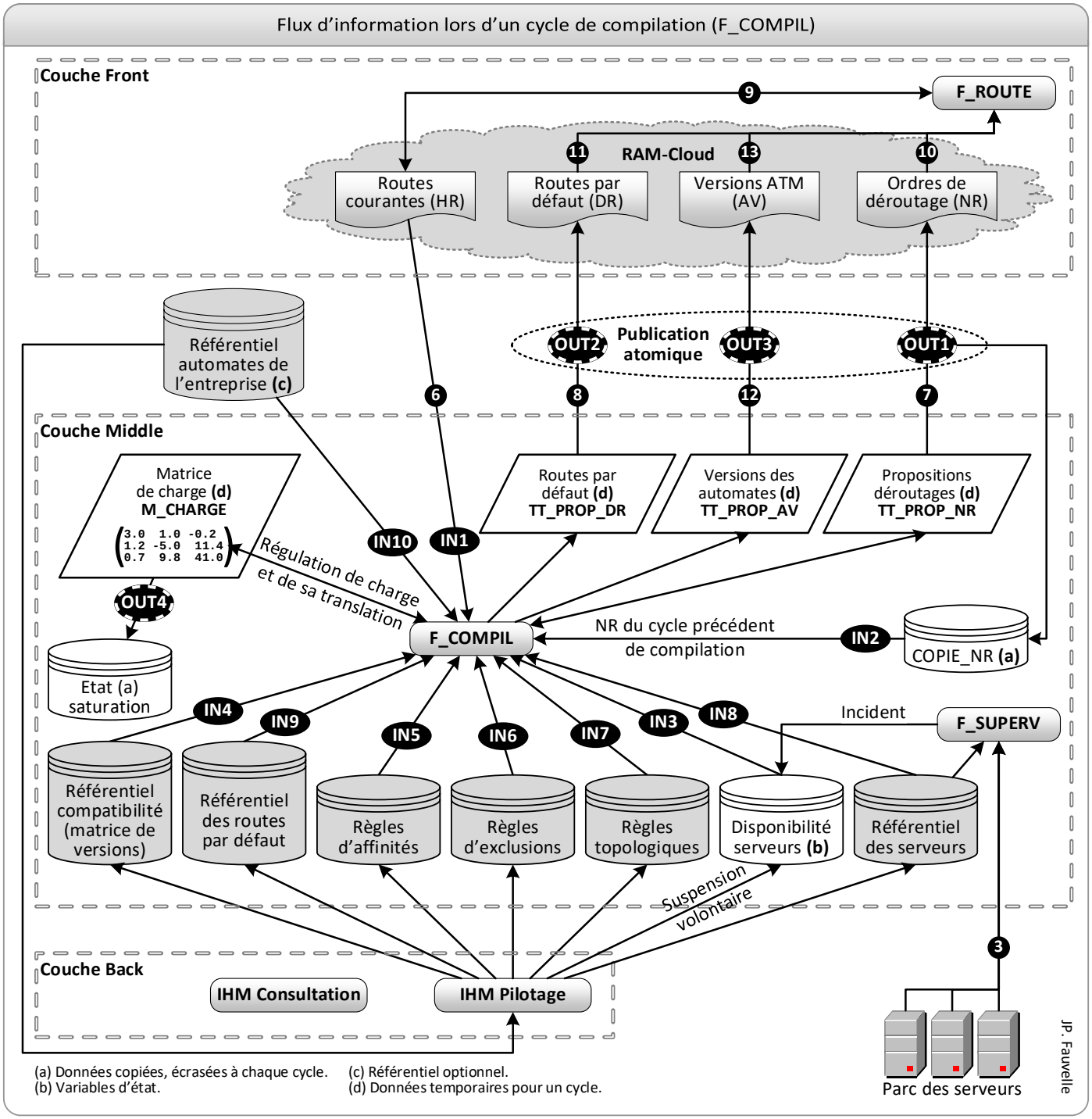

Figure 17 — Flux d'information lors d'un cycle de compilation.

2.1.6.1.3.2 MATRICE DE CHARGE (M_CHARGE)

À ce stade du prologue de compilation, une matrice temporaire de charge nommée **M_CHARGE** est initialisée.

Cette matrice représentée dans le Tableau 13 comprend une ligne pour chaque serveur existant dans le référentiel des serveurs (Tableau 12 et Figure 17 / Flux IN8) et comprend au moins les colonnes suivantes :

- C1 : le serveur, noté S, représenté par sa route « SRV_ADDR : SRV_PORT ».
- C2 : le nombre de routes courantes associées à ce serveur et qui existent actuellement.

  Cette information correspond au nombre d'occurrences de la route C1 existant à l'état enrôlé ou partiellement enrôlé dans la table des routes courantes (Tableau 12 et Figure 17 / Flux IN1).

- C3 : le nombre de déroutages entrant vers ce serveur et qui sont en attente d'exécution par la fonction de routage.

  Cette information correspond pour chaque serveur, aux ordres de déroutages entrants publiés lors du précédent cycle de compilation (Tableau 12 et Figure 17 / Flux IN2) et qui n'ont pas encore été exécutés c.-à-d. qui ne sont pas encore répercutés dans les routes courantes (Flux IN1).

- C4 : le nombre de déroutages sortants de ce serveur et qui sont en attente d'exécution par la fonction de routage.

  Cette information correspond pour chaque serveur, aux ordres de déroutages sortants publiés lors du précédent cycle de compilation (Tableau 12 et Figure 17 / Flux IN2) et qui n'ont pas encore été exécutés c.-à-d. qui ne sont pas encore répercutés dans les routes courantes (Flux IN1).

- C5 : variable d'état CHARGE_REL(S) calculée pour le serveur S spécifié en colonne C1, conformément au procédé décrit au §2.1.6.1.1 :

$$CHARGE\_REL(C1) = \frac{CHARGE(C1)}{P\_CHARGE\_MAX(C1)} = \frac{C2}{P\_CHARGE\_MAX(C1)}$$

  Un résultat supérieur à 1 est ramené à 1.

- C6 : variable d'état TRANS_REL(S) calculée pour le serveur S spécifié en colonne C1, conformément au procédé décrit au §2.1.6.1.1 :

$$TRANS\_REL(C1) = \frac{TRANS(C1)}{P\_TRANS\_MAX(C1)} = \frac{C3 - C4}{P\_TRANS\_MAX(C1)}$$

  Un résultat négatif est ramené à zéro, un résultat supérieur à 1 est ramené à 1.

- C7 : variable d'état SATUR(S) calculée (*) pour le serveur S spécifié en colonne C1, conformément au procédé décrit au §2.1.6.1.1 :

$$SATUR(C1) = \frac{1 * CHARGE(C1) + P\_TRANS\_SURCOUT * TRANS(C1)}{1 + P\_TRANS\_SURCOUT} = \frac{C2 + P\_TRANS\_SURCOUT * (C3 - C4)}{1 + P\_TRANS\_SURCOUT}$$

  (*) *Lors du calcul de SATUR(C1), toute valeur négative produite par TRANS(C1) est ramenée à zéro.*

C1	C2	C3	C4	C5	C6	C7
Serveur	**Nombre de routes courantes vers ce serveur**	**Nombre de déroutages entrants en attente d'exécution**	**Nombre de déroutages sortants en attente d'exécution**	**CHARGE_REL** $\begin{cases} CHARGE\_REL \in \mathbb{R} \\ CHARGE\_REL \in [0,1] \end{cases}$	**TRANS_REL** $\begin{cases} TRANS\_REL \in \mathbb{R} \\ TRANS\_REL \in [0,1] \end{cases}$	**SATUR** $\begin{cases} SATUR \in \mathbb{R} \\ SATUR \in [0,+\infty[\end{cases}$
« Host_x:8001 »	10	4	1	0.17 (17%)	0.39 (39%)	6.12
« Host_y:8080 »	14	2	6	0.28 (28%)	0.00 (0%)	2.87
…	…	…	…	…	…	…

Tableau 13 — Matrice temporaire de charge M_CHARGE (exemple avec données fictives).

Cette matrice temporaire :

1. Est conservée pendant le présent cycle de compilation jusqu'à son épilogue.
2. Est consultée et mise à jour par certaines phases du présent cycle de compilation, toute modification effectuée par une phase étant visible par les phases suivantes du même cycle.
3. Est abandonnée en fin du cycle de compilation.

En fin de cycle de compilation, le contenu de cette matrice peut éventuellement être conservé pour fournir une vision de l'état de charge de la flotte et du parc (Figure 17 / Flux OUT4).

2.1.6.1.3.3 TABLE TEMPORAIRE DES ROUTES PAR DÉFAUT (TT_PROP_DR)

À ce stade du prologue de compilation, une liste temporaire nommée **TT_PROP_DR** est créée, qui sera conservée pendant le présent cycle de compilation jusqu'à son épilogue :

1. La table est initialisée vide.
2. La table est peuplée par la phase 1 du présent cycle de compilation (§2.1.6.1.4).
3. La table est publiée puis abandonnée en fin du présent cycle de compilation.

2.1.6.1.3.4 TABLE TEMPORAIRE DES PROPOSITIONS DE DÉROUTAGES (TT_PROP_NR)

À ce stade du prologue de compilation, une table temporaire nommée **TT_PROP_NR** est créée, qui sera conservée pendant le présent cycle de compilation jusqu'à son épilogue :

1. La table est initialisée vide.
2. La table est peuplée par certaines phases du présent cycle de compilation, toute modification effectuée lors d'une phase étant visible lors des phases suivantes du même cycle de compilation.
3. La table est publiée puis abandonnée en fin du présent cycle de compilation.

2.1.6.1.3.5 TABLE TEMPORAIRE DES VERSIONS AUTOMATES (TT_PROP_AV)

À ce stade du prologue de compilation, une table temporaire nommée **TT_PROP_AV** est créée, qui sera conservée pendant le présent cycle de compilation jusqu'à son épilogue :

1. La table est initialisée vide.
2. La table est peuplée par la phase 1 du présent cycle de compilation (§2.1.6.1.4).
3. La table est publiée puis abandonnée en fin du présent cycle de compilation.

2.1.6.1.4 COMPILATION PHASE 1 : DONNÉES POUR L'ENRÔLEMENT DES AUTOMATES

1. Proposition des routes par défaut :
    a. La fonction F_COMPIL identifie l'ensemble des routes par défaut (Tableau 12 et Figure 17 / Flux IN9) et en exclut les serveurs actuellement statués indisponibles (Flux IN3).
    b. Puis la fonction F_COMPIL **propose** ces routes par défaut (écriture vers TT_PROP_DR) qui ne seront publiées qu'en phase d'épilogue.
2. Proposition de l'unique route doublement par défaut :
    a. La fonction F_COMPIL **propose** la route particulière doublement par défaut « MOST_COMMON_VERSION » (écriture vers TT_PROP_DR) dont la valeur est fixée par configuration via l'IHM de la couche Back.
3. Proposition des versions des automates, s'il existe un référentiel des automates :
    a. La fonction F_COMPIL identifie les couples (atm_id, atm_version) pour les automates connus du référentiel de l'entreprise, s'il existe (Tableau 12 et Figure 17 / Flux IN10).
    b. Puis la fonction F_COMPIL **propose** ces versions automates (écriture vers TT_PROP_AV) qui ne seront publiées qu'en phase d'épilogue.

    La table TT_PROP_AV reste vide s'il n'existe pas de référentiel des automates.

2.1.6.1.5 COMPILATION PHASE 2 : DÉROUTAGE SUR INCIDENT OU ÉVITEMENT D'INCIDENT

Cette phase implémente le déroutage des automates dans deux situations :

1. Lorsqu'un automate est routé vers un serveur incidenté, alors l'automate est hors service.

   La publication puis l'exécution d'un ordre déroutant l'automate vers un serveur disponible, auront pour effet de remettre l'automate en service.

2. Lorsque pour un automate, existe un ordre de déroutage en attente d'exécution dont le serveur cible est incidenté, alors l'exécution dudit ordre aura pour effet soit de mettre l'automate hors service soit de le maintenir hors service s'il tel est déjà le cas.

   La publication puis l'exécution d'un ordre déroutant l'automate vers un serveur disponible, auront pour effet soit d'éviter la mise hors service de l'automate soit de provoquer sa remise en service.

Les flux mentionnés ci-après renvoient au Tableau 12 et à la Figure 17.

2.1.6.1.5.1 AUTOMATES CANDIDATS

La fonction F_COMPIL calcule une liste L_ATM_CANDIDATS comprenant tous les automates affectés par un incident ou sur le point de l'être.

Toute route courante (flux IN1) notée HR comprend un champ HR.atm_id défini et faisant référence à un automate d'identifiant ID. À tout automate d'identifiant ID est éventuellement associée une route courante lui faisant référence via le champ HR.atm_id. S'il existe plusieurs routes courantes faisant référence au même automate, alors seule la plus récente est considérée c.-à-d. celle possédant la plus grande valeur sur le champ HR.req_time.

La liste L_ATM_CANDIDATS est initialisée vide puis peuplée avec l'ensemble des automates dont chaque automate A d'identifiant ID vérifie simultanément toutes les conditions suivantes :

1. Il existe une route courante HR dont le champ HR.atm_id fait référence à l'automate (c.-à-d. HR.atm_id est égale à ID).
2. Ladite route courante HR est effective (c.-à-d. HR.hot_route est différente de « REFUSED »).
3. L'automate est enrôlé (c.-à-d. HR.atm_version est définie).
4. Les conditions du tableau ci-après sont respectées :

Un incident existe : l'automate est actuellement routé vers un serveur incidenté. **ET** **L'incident est durable** : l'automate n'est pas en cours de déroutage ou alors il est en cours de déroutage vers un serveur lui aussi incidenté.	**OU**	**Pas d'incident** : l'automate est routé actuellement vers un serveur non incidenté. **ET** **Mais incident à venir** : l'automate est déjà en cours de déroutage vers un serveur incidenté.

2.1.6.1.5.2 ROUTES POSSIBLES

La liste L_ATM_CANDIDATS utilisée ci-après a été calculée au §2.1.6.1.5.1.

Pour chaque automate A présent dans la liste L_ATM_CANDIDATS, la fonction F_COMPIL calcule une liste L_ROUTES_POSSIBLES(A), éventuellement vide, comprenant toutes les routes possibles pour ledit automate A.

Le calcul de L_ROUTES_POSSIBLES(A) pour un automate A équipé d'un progiciel client dont la version V est définie, et possédant une route courante HR, s'effectue en exécutant les opérations ci-après dans cet ordre :

1. L_ROUTES_POSSIBLES(A) est initialisée avec l'ensemble des routes associées aux serveurs existants (flux IN8).
2. L_ROUTES_POSSIBLES(A) est expurgée des routes associées aux serveurs indisponibles, quel que soit le motif de l'indisponibilité (flux IN3).
3. L_ROUTES_POSSIBLES(A) est expurgée des routes associées aux serveurs qui sont équipés d'un progiciel serveur dont la version n'est pas compatible avec la version V du progiciel client équipant A (flux IN4).
4. Si l'automate A est affecté par au moins une règle active d'affinité vers certains serveurs (flux IN5), alors la liste L_ROUTES_POSSIBLES(A) est expurgée des routes associées aux serveurs autres que ceux référencés par lesdites règles.

   Dans le cas contraire (c.-à-d. s'il n'existe aucune règle active d'affinité affectant A) alors L_ROUTES_POSSIBLES(A) reste inchangée.
5. Si l'automate A est affecté par au moins une règle active d'exclusion de certains serveurs (flux IN6), alors la liste L_ROUTES_POSSIBLES(A) est expurgée des routes associées aux serveurs référencés par lesdites règles.

   Dans le cas contraire (c.-à-d. s'il n'existe aucune règle active d'exclusion affectant A) alors L_ROUTES_POSSIBLES(A) reste inchangée.
6. Si l'automate A est effectivement routé (c.-à-d. HR.hot_route est différente de « REFUSED ») vers un serveur dit « serveur source » noté S1, alors :
   a. Si S1 est actuellement incidenté (flux IN3), alors la liste L_ROUTES_POSSIBLES(A) est expurgée de la route associée à chaque serveur dit « serveur cible » noté S2 pour lequel la matrice de topologie spécifie que les déroutages de S1 vers S2 sont soumis à la condition C_JAMAIS (flux IN7).
   b. Dans le cas contraire (c.-à-d. si le serveur S1 n'est pas incidenté) alors la liste L_ROUTES_POSSIBLES(A) est expurgée de la route associée à chaque serveur dit « serveur cible » noté S2 pour lequel la matrice de topologie spécifie que les déroutages de S1 vers S2 sont soumis à la condition C_JAMAIS ou C_INCIDENT (flux IN7).

   Dans le cas contraire (c.-à-d. si HR.hot_route est égale à « REFUSED ») alors L_ROUTES_POSSIBLES(A) reste inchangée.
7. La liste L_ROUTES_POSSIBLES(A) est expurgée de la route associée à chaque serveur noté S pour lequel au moins l'une des variables d'état CHARGE_REL(S) ou TRANS_REL(S) est saturée (c.-à-d. égale à 1) dans la matrice M_CHARGE (Tableau 13).

2.1.6.1.5.3 DÉROUTAGES PROPOSÉS

La liste L_ATM_CANDIDATS utilisée ci-après a été calculée au §2.1.6.1.5.1.

Les listes L_ROUTES_POSSIBLES utilisées ci-après ont été calculées au §2.1.6.1.5.2.

Pour chaque automate A appartenant à la liste L_ATM_CANDIDATS, tour à tour par ordre croissant du nombre de routes possibles conformément à l'heuristique socio-mimétique (§2.1.6.1.2) :

1. La fonction F_COMPIL choisit dans la liste L_ROUTES_POSSIBLES(A) la route possible, notée R, qui possède dans la matrice M_CHARGE la plus petite valeur de SATUR(S), où S est le serveur associé à R :
   a. S'il existe des routes ex aequo, alors le départage s'effectue sur la plus petite valeur de TRANS_REL(S).

b. S'il existe encore des ex aequo, alors le départage s'effectue sur la plus petite valeur de CHARGE_REL(S).

c. S'il persiste toujours des ex aequo, alors le départage est quelconque, par exemple aléatoire.

2. Puis F_COMPIL propose dans TT_PROP_NR un déroutage noté NR, pour l'automate A d'identifiant ID et de route courante HR :

Champ	←	Valeur	Condition	Remarque
NR.atm_id	←	ID	Inconditionnel	Clé primaire.
NR.next_route	←	R	Il existe une route possible R.	
	←	« REFUSED »	Il n'existe pas de route possible R.	
NR.is_immediate	←	VRAI	La route courante HR est incidentée.	
	←	FAUX	La route courante HR n'est pas incidentée.	
NR.atm_version	←	HR.atm_version	Inconditionnel.	Toujours défini puisque A est enrôlé.

3. Puis F_COMPIL met à jour la matrice M_CHARGE (Tableau 13) afin qu'elle reflète l'effet de ce déroutage proposé s'il était exécuté, comme suit :

a. Si la route actuelle de l'automate A est effective (c.-à-d. si HR.hot_route est différente de « REFUSED ») alors décrémenter la colonne C2 et incrémenter la colonne C4 dans la matrice M_CHARGE pour la ligne correspondant au serveur source c.-à-d. vers lequel l'automate A est actuellement routé.

b. Si le déroutage proposé pour l'automate A est effectif (c.-à-d. si NR.next_route est différente de « REFUSED ») alors incrémenter les colonnes C2 et C3 dans la matrice M_CHARGE pour la ligne correspondant au serveur cible c.-à-d. vers lequel l'automate A doit être dérouté.

c. Dans tous les cas, recalculer conformément au §2.1.6.1.1 les colonnes C5 / C6 / C7 dans la matrice M_CHARGE pour les lignes modifiées ci-dessus.

2.1.6.1.6 COMPILATION PHASE 3 : DÉROUTAGE SUR RÈGLE D'EXCLUSION

Cette phase implémente le déroutage des automates affectés par au moins une règle active d'exclusion.

Les flux mentionnés renvoient au Tableau 12 et à la Figure 17.

2.1.6.1.6.1 AUTOMATES CANDIDATS

La fonction F_COMPIL calcule une liste L_ATM_CANDIDATS comprenant tous les automates affectés par au moins une règle active d'exclusion.

Toute route courante (flux IN1) notée HR comprend un champ HR.atm_id défini et faisant référence à un automate d'identifiant ID. À tout automate d'identifiant ID est éventuellement associée une route courante lui faisant référence via le champ HR.atm_id. S'il existe plusieurs routes courantes faisant référence au même automate, alors seule la plus récente est considérée c.-à-d. celle possédant la plus grande valeur sur le champ HR.req_time.

La liste L_ATM_CANDIDATS est initialisée vide puis peuplée avec l'ensemble des automates dont chaque automate A d'identifiant ID vérifie simultanément toutes les conditions suivantes :

1. Il existe une route courante HR dont le champ HR.atm_id fait référence à l'automate (c.-à-d. HR.atm_id est égale à ID).
2. Ladite route courante HR est effective (c.-à-d. HR.hot_route est différente de « REFUSED »).
3. L'automate est enrôlé (c.-à-d. HR.atm_version est définie).
4. Aucun déroutage pour cet automate n'a été proposé dans TT_PROP_NR par une phase précédente du cycle de compilation en cours.
5. Les conditions du tableau ci-après sont respectées :

Il existe une exclusion : l'automate est routé actuellement vers un serveur affecté par une règle d'exclusion active. **ET** **L'exclusion est durable** : l'automate n'est pas en cours de déroutage ou alors il est en cours de déroutage vers un serveur affecté par une règle d'exclusion active.	**OU**	**Il n'existe pas d'exclusion** : l'automate est routé actuellement vers un serveur non affecté par une règle d'exclusion active. **ET** **Une exclusion est imminente** : l'automate est en cours de déroutage vers un serveur affecté par une règle d'exclusion active.

2.1.6.1.6.2 ROUTES POSSIBLES

La fonction F_COMPIL calcule la liste L_ROUTES_POSSIBLES(A) en utilisant le même algorithme déjà spécifié pour la phase 2 de la compilation (§2.1.6.1.5.2) avec les différences suivantes :

1. La liste L_ATM_CANDIDATS utilisée par l'algorithme est celle calculée au §2.1.6.1.6.1.

2.1.6.1.6.3 DÉROUTAGES PROPOSÉS

La fonction F_COMPIL calcule des déroutages en utilisant le même algorithme déjà spécifié pour la phase 2 de la compilation (§2.1.6.1.5.3) avec les différences suivantes :

1. La liste L_ATM_CANDIDATS utilisée par l'algorithme est celle calculée au §2.1.6.1.6.1.
2. Les listes L_ROUTES_POSSIBLES utilisées par l'algorithme sont celles calculées au §2.1.6.1.6.2.
3. Le champ NR.is_immediate de tout déroutage NR proposé doit être positionné à FAUX.

2.1.6.1.7 COMPILATION PHASE 4 : DÉROUTAGE SUR RÈGLE D'AFFINITÉ

Cette phase implémente le déroutage des automates affectés par au moins une règle active d'affinités.

Les flux mentionnés renvoient au Tableau 12 et à la Figure 17.

2.1.6.1.7.1 AUTOMATES CANDIDATS

La fonction F_COMPIL calcule une liste L_ATM_CANDIDATS comprenant tous les automates affectés par au moins une règle active d'affinité.

Toute route courante (flux IN1) notée HR comprend un champ HR.atm_id défini et faisant référence à un automate d'identifiant ID. À tout automate d'identifiant ID est éventuellement associée une route courante lui faisant référence via le champ HR.atm_id. S'il existe plusieurs routes courantes faisant référence au même automate, alors seule la plus récente est considérée c.-à-d. celle possédant la plus grande valeur sur le champ HR.req_time.

La liste L_ATM_CANDIDATS est initialisée vide puis peuplée avec l'ensemble des automates dont chaque automate A d'identifiant ID vérifie simultanément toutes les conditions suivantes :

1. Il existe une route courante HR dont le champ HR.atm_id fait référence à l'automate (c.-à-d. HR.atm_id est égale à ID).
2. Ladite route courante HR est effective (c.-à-d. HR.hot_route est différente de « REFUSED »).
3. L'automate est enrôlé (c.-à-d. HR.atm_version est définie).
4. Aucun déroutage pour cet automate n'a été proposé dans TT_PROP_NR par une phase précédente du cycle de compilation en cours.
5. Les conditions du tableau ci-après sont respectées :

Il existe une violation d'affinité : l'automate est affecté par une règle d'affinité et sa route actuelle viole ladite affinité. **ET** **Elle est durable** : l'automate n'est pas en cours de déroutage, ou alors il est en cours de déroutage et sa route à venir viole une règle d'affinité affectant l'automate.	**OU**	**Il n'existe pas de violation d'affinité** : l'automate est affecté par une règle d'affinité mais sa route actuelle s'y conforme. **ET** **Violation à venir** : l'automate est en cours de déroutage et la route à venir viole une règle d'affinité affectant l'automate.

2.1.6.1.7.2 ROUTES POSSIBLES

La fonction F_COMPIL calcule la liste L_ROUTES_POSSIBLES(A) en utilisant le même algorithme déjà spécifié pour la phase 2 de la compilation (§2.1.6.1.5.2) avec les différences suivantes :

1. La liste L_ATM_CANDIDATS utilisée par l'algorithme est celle calculée au §2.1.6.1.7.1.

2.1.6.1.7.3 DÉROUTAGES PROPOSÉS

La fonction F_COMPIL calcule des déroutages en utilisant le même algorithme déjà spécifié pour la phase 2 de la compilation (§2.1.6.1.5.3) avec les différences suivantes :

1. La liste L_ATM_CANDIDATS utilisée par l'algorithme est celle calculée au §2.1.6.1.7.1.
2. Les listes L_ROUTES_POSSIBLES utilisées par l'algorithme sont celles calculées au §2.1.6.1.7.2.
3. Le champ NR.is_immediate de tout déroutage proposé NR doit être positionné à FAUX.

2.1.6.1.8 COMPILATION PHASE 5 : RECYCLAGE DES ROUTES REFUSÉES

Cette phase implémente le recyclage des automates dont la route courante actuelle n'est pas effective (c.-à-d. dont la route courante prend la valeur « REFUSED ») et dont les requêtes sont refusées à ce titre.

Les flux mentionnés renvoient au Tableau 12 et à la Figure 17.

2.1.6.1.8.1 AUTOMATES CANDIDATS

La fonction F_COMPIL calcule une liste L_ATM_CANDIDATS comprenant tous les automates dont la route courante n'est pas effective.

Toute route courante (flux IN1) notée HR comprend un champ HR.atm_id défini et faisant référence à un automate d'identifiant ID. À tout automate d'identifiant ID est éventuellement associée une route courante lui faisant référence via le champ HR.atm_id. S'il existe plusieurs routes courantes faisant référence au même automate, alors seule la plus récente est considérée c.-à-d. celle possédant la plus grande valeur sur le champ HR.req_time.

La liste L_ATM_CANDIDATS est initialisée vide puis peuplée avec l'ensemble des automates dont chaque automate A d'identifiant ID vérifie simultanément toutes les conditions suivantes :

1. Il existe une route courante HR dont le champ HR.atm_id fait référence à l'automate (c.-à-d. HR.atm_id est égale à ID).
2. Ladite route courante HR n'est pas effective (c.-à-d. HR.hot_route est égale à « REFUSED »).
3. L'automate est enrôlé (c.-à-d. HR.atm_version est définie).
4. Aucun déroutage pour cet automate n'est en attente d'exécution (flux IN2).
5. Aucun déroutage pour cet automate n'a été proposé dans TT_PROP_NR par une phase précédente du cycle de compilation en cours.

2.1.6.1.8.2 ROUTES POSSIBLES

La fonction F_COMPIL calcule la liste L_ROUTES_POSSIBLES(A) en utilisant le même algorithme déjà spécifié pour la phase 2 de la compilation (§2.1.6.1.5.2) avec les différences suivantes :

1. La liste L_ATM_CANDIDATS utilisée par l'algorithme est celle calculée au §2.1.6.1.8.1.

2.1.6.1.8.3 DÉROUTAGES PROPOSÉS

La fonction F_COMPIL calcule des déroutages en utilisant le même algorithme déjà spécifié pour la phase 2 de la compilation (§2.1.6.1.5.3) avec les différences suivantes :

1. La liste L_ATM_CANDIDATS utilisée par l'algorithme est celle calculée au §2.1.6.1.8.1.
2. Les listes L_ROUTES_POSSIBLES utilisées par l'algorithme sont celles calculées au §2.1.6.1.8.2.
3. Le champ NR.is_immediate de tout déroutage NR proposé doit être positionné à FAUX.

2.1.6.1.9 COMPILATION PHASE 6 : ÉQUILIBRAGE DE CHARGE ET DE TRANSLATION DE CHARGE

Cette phase implémente l'équilibrage de la charge et de la translation de charge entre les serveurs constituant le parc, ce qui consiste à dérouter un nombre limité et paramétrable d'automates à chaque cycle de compilation.

Les flux mentionnés renvoient au Tableau 12 et à la Figure 17.

2.1.6.1.9.1 PIVOTS CANDIDATS

La fonction F_COMPIL calcule une liste nommée L_PIVOTS_CANDIDATS comprenant tous les couples de serveurs possibles, chaque couple pivot étant noté CP = (source S1, cible S2), pour lesquels il est possible de dérouter des automates depuis le serveur source S1 vers le serveur cible S2.

Pour ce faire, la fonction F_COMPIL identifie tous les couples pivots notés CP = (source S1, cible S2) vérifiant simultanément les conditions suivantes :

1. Les serveurs S1 et S2 sont distincts.
2. Les serveurs S1 et S2 sont disponibles (flux IN3).
3. Les serveurs S1 et S2 sont équipés soit de la même version du progiciel serveur, soit de versions V1 et V2 parfaitement compatibles c.-à-d. telles que toute version du progiciel client qui est compatible avec V1 est également compatible avec V2 et inversement (flux IN8).
4. Il existe une règle topologique spécifiant que le déroutage depuis S1 vers S2 est autorisé inconditionnellement donc en particulier lorsque S1 n'est pas incidenté, c'est-à-dire en d'autres termes que le déroutage depuis S1 vers S2 est autorisé sous la condition C_TOUJOURS (flux IN7).
5. La variable d'état CHARGE_REL(S2) dans la matrice M_CHARGE est inférieure à 1.

Cette condition garantit qu'un serveur cible surchargé en nombre d'automates ne recevra pas une charge supplémentaire.

6. La variable d'état TRANS_REL(S2) dans la matrice M_CHARGE est inférieure à 1.

   Cette condition garantit qu'un serveur cible surchargé en nombre de déroutages entrants ne recevra pas une charge supplémentaire.

7. La variable d'état CHARGE_REL(S1) dans la matrice M_CHARGE est supérieure à un paramètre configurable nommé P_SEUIL_EQUILIBRAGE :

$$CHARGE\_REL(S1) > P\_SEUIL\_EQUILIBRAGE \quad \begin{cases} P\_SEUIL\_EQUILIBRAGE \in \mathbb{R} \\ P\_SEUIL\_EQUILIBRAGE \in]0,1[\end{cases}$$

   Cette condition garantit qu'un serveur source peu chargé en nombres d'automates ne fera pas l'objet d'un déchargement inutile.

8. Le pivot source S1 est plus saturé que le pivot cible S2, et plus précisément l'écart relatif de saturation entre eux, noté ECART_REL_PIVOT(S1, S2), est supérieur à un paramètre configurable nommé P_GAIN_EQUILIBRAGE :

$$ECART\_REL\_PIVOT(S1,S2) = \frac{SATUR(S1) - SATUR(S2)}{SATUR(S1)} > 0$$

$$ECART\_REL\_PIVOT(S1,S2) > P\_GAIN\_EQUILIBRAGE \quad \begin{cases} P\_GAIN\_EQUILIBRAGE \in \mathbb{R} \\ P\_GAIN\_EQUILIBRAGE \in]0,1[\end{cases}$$

   Cette condition garantit que la translation de charge entre le serveur source et le serveur cible représente un intérêt suffisant.

   À titre d'exemple, fixer le paramètre P_GAIN_EQUILIBRAGE à la valeur 0.15 a pour effet d'exclure de la liste L_PIVOTS_CANDIDATS tous les couples (source, cible) pour lesquels la saturation du serveur cible n'est pas inférieure d'au moins 15% à celle du serveur source.

Puis la fonction F_COMPIL ordonne cette liste L_PIVOTS_CANDIDATS selon les valeurs décroissantes de ECART_REL_PIVOT(source S1, cible S2). En d'autres termes, plus un serveur pivot source est saturé par rapport à son serveur pivot cible associé, plus le couple pivot noté CP = (source S1, cible S2) est proche du début de ladite liste ordonnée.

2.1.6.1.9.2 AUTOMATES CANDIDATS

Puis pour chaque couple pivot noté CP = (source S1, cible S2) compris dans la liste L_PIVOTS_CANDIDATS, la fonction F_COMPIL calcule une liste L_ATM_CANDIDATS(CP) comprenant tous les automates qui sont actuellement routés vers S1 et qui vérifient les conditions suivantes :

1. Aucun ordre de déroutage en attente d'exécution n'existe pour lesdits automates (flux IN2).
2. Aucun ordre de déroutage n'a été proposé dans TT_PROP_NR pour lesdits automates par une phase précédente du cycle de compilation en cours.
3. Aucune règle d'affinité ou d'exclusion n'interdit de dérouter lesdits automates vers S2 (flux IN5 et IN6).

Le résultat est constitué, pour chaque couple pivot noté CP = (source S1, cible S2) compris dans L_PIVOTS_CANDIDATS, d'une liste L_ATM_CANDIDATS(CP) comprenant tous les automates actuellement routés vers S1 et dont le déroutage est possible vers S2.

2.1.6.1.9.3 ROUTES POSSIBLES ET DÉROUTAGES PROPOSÉS

La fonction F_COMPIL propose des ordres de déroutages en exécutant le mode opératoire suivant :

1. Pour chaque couple pivot noté CP = (source S1, cible S2) compris dans la liste L_PIVOTS_CANDIDATS ordonnée :
   A. Pour chaque automate A d'identifiant ID et compris dans la liste L_ATM_CANDIDATS(CP) :
      i. Si le nombre de déroutages déjà proposés par la présente phase lors du présent cycle de compilation, excède un paramètre configurable nommé P_QUOTA_EQUILIBRAGE, alors le traitement s'achève au point 2 ci-après.

         En effet, chaque cycle de compilation permet de dérouter un nombre limité d'automates, l'équilibrage global de la charge s'étalant dans le temps au fil des cycles.

         Compte tenu de l'ordonnancement de la liste L_PIVOTS_CANDIDATS, les déroutages effectués sur les premiers couples de ladite liste sont les plus contributifs à l'équilibrage de la charge.

      ii. Si le déroutage de l'automate A depuis le serveur source S1 vers le serveur cible S2 aurait pour effet que SATUR(S2) devienne supérieur ou égal à SATUR(S1), alors le potentiel de rééquilibrage est épuisé pour le couple pivot CP, ce qui a pour effet :
         a. L'automate A ne doit pas être dérouté vers S2.
         b. Le traitement se poursuit directement pour le prochain couple pivot CP (point 1 ci-avant).

         Ce procédé évite l'inversion des rôles pivot source / pivot cible d'un cycle à l'autre de compilation c.-à-d. évite l'oscillation autour du point d'équilibre de la charge.

      iii. Dans le cas contraire du point précédent (ii), l'automate A est déroutable vers S2 ce qui a pour effet :
         a. La fonction F_COMPIL propose dans TT_PROP_NR, ce déroutage NR vers le serveur cible S2 pour l'automate A d'identifiant ID et de route courante HR :

Champ	←	Valeur	Condition	Remarque
NR.atm_id	←	ID	Inconditionnel	Clé primaire.
NR.next_route	←	S2	Inconditionnel	
NR.is_immediate	←	FAUX	Inconditionnel	
NR.atm_version	←	HR.atm_version	Inconditionnel	Toujours défini puisque A est enrôlé.

         b. À chaque déroutage proposé, la fonction F_COMPIL met à jour la matrice M_CHARGE (Tableau 13) afin qu'elle reflète l'effet du déroutage s'il était exécuté, comme suit :
            1er. Décrémenter la colonne C2 et incrémenter la colonne C4 dans la matrice M_CHARGE pour la ligne correspondant au serveur pivot source.
            2e. Incrémenter les colonnes C2 et C3 dans la matrice M_CHARGE pour la ligne correspondant au serveur pivot cible.
            3e. Recalculer conformément au §2.1.6.1.1 les colonnes C5 / C6 / C7 dans la matrice M_CHARGE pour les deux lignes modifiées ci-dessus.
2. La fonction F_COMPIL propose ces déroutages de rééquilibrage dans TT_PROP_NR, qui ne seront publiés qu'en phase d'épilogue.

2.1.6.1.10 COMPILATION : ÉPILOGUE

Au cours du présent cycle de compilation, la fonction F_COMPIL a **proposé** des déroutages dans TT_PROP_NR, des routes par défaut dans TT_PROP_DR, et des versions automates dans TT_PROP_AV.

Dans cette phase épilogue qui termine le processus de compilation, la fonction F_COMPIL **publie** les routes proposées, en exécutant les opérations suivantes dans cet ordre :

1. F_COMPIL effectue un nettoyage des ordres de déroutage proposés, en supprimant tout déroutage proposé inutile c.-à-d. dont la nouvelle route proposée correspond à la route courante (flux IN1).
2. F_COMPIL **publie** de façon atomique les ordres de déroutages depuis TT_PROP_NR vers le Front à destination de la fonction F_ROUTE (Tableau 12 et Figure 17 / Flux OUT1).

   Une copie des ordres de déroutage est conservée par le compilateur (Figure 17 / COPIE_NR) ce qui permettra à F_COMPIL, lors du prochain cycle de compilation, de connaitre les déroutages qui sont toujours en attente d'exécution sur le Front, par recoupement avec les routes courantes.
3. F_COMPIL **publie** de façon atomique les routes par défaut depuis TT_PROP_DR vers le Front à destination de la fonction F_ROUTE (Tableau 12 et Figure 17 / Flux OUT2).
4. F_COMPIL **publie** de façon atomique les versions des automates depuis TT_PROP_AV vers le Front à destination de la fonction F_ROUTE (Tableau 12 et Figure 17 / Flux OUT3).

2.1.6.2 COMPLEXITÉ ALGORITHMIQUE DE LA FONCTION F_COMPIL

Phase	**Étape**	**Complexité temporelle algorithmique** **F : taille de la Flotte (nombre total d'automates)** **P : taille du parc (nombre total de serveurs)**
Phase 1		= o(1)
Phase 2	Calculer L_ATM_CANDIDATS :	= o(F)
	Calculer L_ROUTES_POSSIBLES :	= o(P * F)
	Ordonner L_ATM_CANDIDATS (quicksort) :	= o(F * $\log_{10}$(F))
	Proposer les routes :	= o(P * F)
Phase 3	Idem phase 2	
Phase 4	Idem phase 2	
Phase 5	Idem phase 2	
Phase 6	Calculer L_PIVOTS_CANDIDATS :	= o(P^2)
	Ordonner L_PIVOTS_CANDIDATS (quicksort) :	= o(P^2 * $\log_{10}$(P^2))
	Calculer L_ATM_CANDIDATS :	= o(L_PIVOTS_CANDIDATS * F) = o(P^2 * F)
	Proposer des routes :	Négligeable car le paramètre P_QUOTA_EQUILIBRAGE limite le nombre de déroutages par cycle de compilation.
Complexité globale de F_COMPIL :		**= o((4+8P+P^2)*F + 4*F*$\log_{10}$(F))**

2.2 PÉRIMÈTRE DE L'INVENTION

2.2.1 DOMAINE TECHNIQUE DE L'INVENTION

L'invention est un composant logiciel offrant des fonctions de gestion de flotte et de routage applicatif intelligent des flux à session des automates bancaires, mais éventuellement utilisable dans un contexte autre que la monétique et pour un protocole autre que HTTP/S.

2.2.2 CARACTÉRISTIQUES ESSENTIELLES ET SECONDAIRES DE L'INVENTION

Le tableau ci-après reprend les caractéristiques de la solution présentées aux §2.1.1 pour y distinguer les essentielles et les secondaires.

Réf.	Caractéristiques de l'invention	Importance
1	Agit en mandataire inverse (*reverse proxy*).	Secondaire
2	S'adapte facilement à tout progiciel client / serveur.	**Essentiel**
3	S'adapte à tout parc de serveurs via paramétrage de règles topologiques.	**Essentiel**
4	Accepte un nombre illimité d'automates grâce à une capacité de mise à l'échelle horizontale à chaud.	**Essentiel**
5	Supporte une segmentation logique multicritères du parc.	**Essentiel**
6	Prend en compte la multiplicité des versions déployées du progiciel entre automates et serveurs.	**Essentiel**
7	Permet l'enrôlement immédiat des automates sans besoin d'un référentiel optionnel des automates.	**Essentiel**
8	Améliore le taux de disponibilité de la flotte d'automates via une heuristique socio-mimétique.	**Essentiel**
9	Optimise l'utilisation du parc de serveurs en distribuant et plafonnant dynamiquement la charge.	**Essentiel**
10	Optimise l'utilisation du parc de serveurs en régulant la translation de charge et en intégrant l'éventuelle surconsommation de ressources par le progiciel lors du traitement des primo-requêtes post-déroutage.	**Essentiel**
11	Contient puis régule la propagation cascadée des surcharges liées aux incidents en masse.	**Essentiel**
12	Permet une vision en temps réel de l'état de la flotte d'automates, du parc de serveurs, des déroutages en cours et de leurs causes.	**Essentiel**
13	Une évolution simple permettrait d'intégrer une fonction dite « d'injection marketing ».	Secondaire

Tableau 14 — Caractéristiques essentielles et secondaires de l'invention.

2.2.3 COMPARAISON DE L'INVENTION AVEC UN PRODUIT PHARE DU MARCHÉ

Le produit HA-Proxy est une référence en termes de solution de routage de flux avec équilibrage de la charge.

Le tableau ci-après reprend les caractéristiques de la solution inventive présentées aux §2.1.1 et identifie celles supportées par HA-Proxy.

Réf.	Caractéristiques de l'invention	Supporté par HA-Proxy
1	Agit en mandataire inverse (*reverse proxy*).	Oui
2	S'adapte facilement à tout progiciel client / serveur.	Non
3	S'adapte à tout parc de serveurs via paramétrage de règles topologiques.	Non
4	Accepte un nombre illimité d'automates grâce à une capacité de mise à l'échelle horizontale à chaud.	Oui

5	Supporte une segmentation logique multicritères du parc.	Non
6	Prend en compte la multiplicité des versions déployées du progiciel entre automates et serveurs.	Non
7	Permet l'enrôlement immédiat des automates sans besoin d'un référentiel optionnel des automates.	Non
8	Améliore le taux de disponibilité de la flotte d'automates via une heuristique socio-mimétique.	Non
9	Optimise l'utilisation du parc de serveurs en distribuant et plafonnant dynamiquement la charge.	Oui
10	Optimise l'utilisation du parc de serveurs en régulant la translation de charge et en intégrant l'éventuelle surconsommation de ressources par le progiciel lors du traitement des primo-requêtes post-déroutage.	Non
11	Contient puis régule la propagation cascadée des surcharges liées aux incidents en masse.	Non
12	Permet une vision en temps réel de l'état de la flotte d'automates, du parc de serveurs, des déroutages en cours et de leurs causes.	Non
13	Une évolution simple permettrait d'intégrer une fonction dite « d'injection marketing ».	Non

Tableau 15 — Comparaison des caractéristiques entre l'invention et HA-Proxy.

2.2.4 APPLICATIONS PRIVILÉGIÉES

Les domaines et applications privilégiés sont les suivants :

- Domaine monétique - Flotte de DAB, GAB.
- Domaine billettique - Flotte de bornes d'achats des titres de transport ferroviaires et aériens.
- Domaine ludique - Flotte de bornes de jeux (PMU, Française des Jeux, etc.).
- Domaine santé - Flotte de postes clients à partir desquels des professionnels de santé utilisent des applications notamment pour accéder au dossier médical personnel des patients.