Sendmail X – la nouvelle génération de sendmail

José-Marcio Martins da Cruz
Ecole des Mines de Paris – Centre de Calcul
60, bd Saint Michel
75275 – Paris
Jose-Marcio.Martins@ensmp.fr

Résumé

sendmail est le logiciel serveur de messagerie le plus ancien et probablement le plus utilisé. Depuis plus de vingt ans il est livré en standard avec pratiquement toutes les variantes du système d'exploitation UNIX. La dernière version de sendmail date déjà de plus de dix ans. Cette version est toujours suffisante pour satisfaire les besoins de la plupart des serveurs de messagerie. Néanmoins, sa flexibilité est la cause et la conséquence de sa complexité de mise en oeuvre et administration. Depuis 2001, une nouvelle version est en cours de développement par Claus Assmann. Cette nouvelle version est radicalement différente de la précédente et a pour ambition de satisfaire les besoins des utilisateurs, avec efficacité et sécurité, pour encore les dix prochaines années. Ce papier rappelle l'historique de sendmail, les points forts et les faiblesses de la version actuelle, puis présente la nouvelle version.

Mots clefs

Serveurs de messagerie, MTA, sendmail X

1. Introduction

L'histoire de sendmail se confond avec celle de la messagerie électronique. L'ancêtre de sendmail a été *delivermail* [1], écrit par Eric Allman alors qu'il était encore étudiant à l'Université de Berkeley en 1979. C'était un programme dont le but premier était l'interconnexion des messageries placées dans des réseaux différents : UUCP (Unix to Unix Communication Protocol), ARPAnet et BerkNet. d*elivermail* était livré avec les versions 4.0 et 4.1 de BSD Unix. Sa principale faiblesse était la configuration (et les tables de routage) définies dans le code du programme.

Vers 1980, la migration d'ARPAnet de NCP (Network Control Protocol) vers TCP (Transport Control Protocol), ainsi que la modification dans l'organisation du espace de nommage (plat vers hiérarchique) et l'apparition du protocole SMTP (Simple Mail Transfer Protocol) ont fait que *delivermail* n'était plus suffisamment flexible pour suivre l'évolution de la messagerie. Eric Allman a alors commencé le développement de sendmail avec l'idée d'avoir un logiciel avec la souplesse nécessaire pour suivre l'évolution vu que, à l'époque, rien n'assurait la stabilité des différents environnements.

Eric Allman s'étant détourné du développement de ce programme pendant une longue période, d'autres versions sont apparues pour combler des besoins ponctuels. Par exemple, Lennart Lovstrand (Université de Linkoping – Sweden) a créé la version IDA de la version 5 de sendmail, dont les améliorations principales étaient l'utilisation d'une base de données (dbm) ainsi que des modifications dans la réécriture des entêtes et des enveloppes. Neil Rickert et Paul Pomes ont donné suite à l'évolution de sendmail IDA. Paul Vixie a travaillé sur la version KJS (King's James Sendmail) dont le but principal était plutôt l'optimisation du code. Des constructeurs, en particulier Sun et Hewlett-Packard, ont, eux aussi contribué au développement de sendmail, selon leurs besoins. Sun Microsystems a ajouté, entre autres, le support de *nis* et *nisplus*, tandis que Hewlett-Packard a contribué avec le support 8BITMIME.

En 1992, Eric Allman a regroupé les travaux dispersés et a commencé à travailler sur la version 8 de sendmail qui est la base de la version actuelle. Le point important à retenir est l'environnement existant à l'époque : Internet n'était pas encore ouverte aux grand public, les menaces n'étaient pas encore aussi présentes, les protocoles et normes réseaux n'étaient pas encore stables et certaines caractéristiques des systèmes d'exploitation étaient aussi en évolution (POSIX threads...). Ces points sont essentiels et expliquent les choix pris dans le développement de la version 8 de sendmail.

Au départ, la configuration de sendmail se faisait, avec beaucoup de difficulté, directement sur le fichier de configuration *sendmail.cf*, et c'est sûrement ce qui a fait que sendmail est connu comme un logiciel dont la configuration n'est pas conviviale. Vers 1991, Pierre David et Jacky Thibault ont créé le kit de configuration de Jussieu [2], permettant de masquer la syntaxe difficile du fichier de configuration : l'utilisateur se limitait à activer ou pas certaines fonctionnalités ou attribuer des valeurs à certaines variables sur un fichier qui était, en réalité, un script écrit en shell de Bourne. Quelque temps après est apparu le kit de configuration m4 de sendmail, permettant au utilisateur de décrire la configuration de sendmail, comme dans le kit de Jussieu, mais avec des macros m4. Neil Rickert a été un de ceux qui ont le plus contribué à ce kit.

En 1999, après la sortie de la version 8.9, Eric Allman crée la société Sendmail, Inc. La création de cette entreprise fait apparaître, à nouveau, deux branches de développement.

La branche commerciale a été prise en charge principalement par Greg Shapiro, tandis que la partie ouverte a été prise en charge par Claus Assmann. Ce dernier a été embauché par Sendmail. Inc. Pour s'occuper exclusivement de la version « open source ».

Au départ, la principale différence entre les deux branches était l'interface de configuration bien plus conviviale dans la version commerciale.

La configuration de sendmail se faisait, avec beaucoup de difficulté, directement sur le fichier sendmail.cf.

Depuis, les changements importants depuis ont été l'intégration de STARTTLS et AUTH, des aspects sécuritaires (en particulier la possibilité de faire tourner le

daemon sous une identité autre que `root`), puis l'interface milter[1] permettant d'utiliser un module de filtrage externe. Les autres changements concernent surtout des bugs, des failles de sécurité trouvées ou des évolutions mineures.

La version actuelle de sendmail est la 8.13.4. La nouvelle version majeure, objet de cette présentation, est en état beta au moment de l'écriture de ce papier. Cela signifie que, sauf imprévu, des nouvelles fonctionnalités ne seront pas ajoutées avant sa sortie officielle. Dans la série 8, la version 8.14 est en préparation, mais elle ne fait que consolider les évolutions passées et/ou corriger des problèmes rencontrés.

Vers 1997, d'autres MTAs (Mail Transport Agent) sont apparus. Jusque là, sendmail avait le quasi-monopole des logiciels serveurs de messagerie. Certains ont largement utilisé les faiblesses (justifiées ou exagérées) de sendmail [3] pour trouver leur place. En particulier, les points où sendmail a été le plus critiqué sont des failles de sécurité, le fichier de configuration inintelligible par un humain et parfois les limitations dans ses performances. *Postfix, en particulier,* a apporté une facilité de configuration, une modularité et une conception axée sur la sécurité, qui sont des qualités indéniables.

Et pourtant, même si nombreux sont les utilisateurs qui sont passés à des logiciels concurrents, nombreux sont les irréductibles qui l'utilisent encore. Et en réalité, les besoins de la majorité des serveurs de messagerie sont couverts par la version de sendmail actuelle. Aussi pratiquement toutes les distributions d'Unix modernes sont toujours livrées avec sendmail.

Depuis trois ans, une nouvelle version de sendmail (sendmail X) est en cours de préparation. La conception et le développement ont été réalisés par Claus Assmann, aidé par un groupe d'experts connus comme par exemple, Eric Allman, Gregory Shapiro, Brian Costales, John Beck et Neil Rickert.

Les objectifs sont ambitieux : créer un MTA moderne, performant et surtout avoir une nouvelle durée de vie de 10 ans, tout en profitant, au fur et à mesure, de l'avancée technologique matériel et des systèmes d'exploitation.

Une version déjà utilisable et déjà en production dans certains sites existe déjà. Il est temps d'en parler.

2. sendmail 8 : structure, faiblesses et atouts

Comme nous avons vu, les choix effectués lors de la conception de sendmail sont le résultat de l'environnement de la messagerie électronique à l'époque.

Le fichier de configuration de sendmail est, à la fois, un point fort et un point faible. Il est constitué d'une partie permettant d'attribuer des valeurs aux différents paramètres de fonctionnement et d'une partie contenant des « *règles* ». Ces règles constituent, en réalité, un programme informatique écrit dans un langage qui est à la fois complexe et strict. Le binaire sendmail lui-même ne fait qu'évaluer les valeurs des variables liées à chaque message – les décisions étant définies dans les règles. Ainsi, malgré sa complexité, ces règles présentent l'avantage de rendre sendmail extrêmement flexible : pour modifier le comportement de sendmail, il suffit de modifier les règles, sans besoin de régénérer le code binaire. De même, les modifications apportées sur une version sont valables sur une version future, sans aucune modification du code source en C. Rappelons que la flexibilité était un critère majeur à l'époque où sendmail 8 a été créé. Ce besoin n'ayant actuellement plus cours, la possibilité de définir des règles de traitement n'existera pas dans le fichier de configuration de sendmail X.

Un autre point fort de sendmail est la qualité du code source de sendmail, critère souvent négligé. Le style de programmation très rigoureux [4] rend le code source homogène et lisible facilement. Cette rigueur fait que la génération de la version binaire se fasse sans erreurs ou avertissements, sur une gamme très variée de plateformes (récentes ou pas), systèmes d'exploitation et compilateurs. C'est aussi cette rigueur de programmation qui a fait que sendmail puisse être maintenu et a pu évoluer pendant plus de 10 ans.

Les problèmes de sécurité de sendmail ont été largement (et excessivement) critiqués. A l'époque, l'Internet ne vivait pas sous la menace permanente, comme c'est le cas actuellement. De même, la programmation avec souci de sécurité n'était pas encore aussi développée. Par exemple, les routines permettant la copie et concaténation de chaînes de caractères avec contrôle de contour (`strlcat` et `strlcpy`) ne sont apparues qu'après 1996, proposées par Todd Miller et Theo de Raadt [5] sur OpenBSD. Ces routines, qui ne sont pas disponibles sur certains systèmes d'exploitation, servent à résoudre une des causes les plus fréquentes des failles de débordement de mémoire tampon.

Ainsi, des failles de sécurité sont apparues sur sendmail 8 et apparaissent encore, mais elles deviennent rares et sont toujours corrigées très rapidement. Il n'est pas étonnant que l'on puisse trouver des failles de sécurité sur un produit qui a été conçu il y a plus de dix ans. Sans nier l'existence des failles de sécurité trouvées, on peut aussi remarquer que, sauf dans les cas des serveurs utilisant encore de très vieilles versions, l'exploitation de ces failles reste un phénomène marginal et bien moins fréquent que les failles, par exemple, que l'on trouve sur les serveurs Web.

sendmail est un logiciel monolithique : il contient, dans un seul fichier exécutable, tous les modules dont a besoin un serveur de messagerie : un client SMTP, un serveur SMTP, un gestionnaire de file d'attente...

Un processus sendmail serveur est démarré et, lorsqu'il y a une demande de service, il se duplique (`fork(2)`) et le processus fils se charge de traiter la requête en question. Ce mode de fonctionnement n'est pas efficace lorsqu'on souhaite traiter des débits importants avec beaucoup de concurrence, à cause de la fréquence importante de changements de contexte [6]. Mais à l'époque où sendmail 8 a été conçu, aucun modèle de threads n'était pas encore spécifié et aucune implémentation n'était suffisamment validée pour être utilisée en production.

Même actuellement, aucun des MTAs concurrents directs de sendmail n'utilise encore le modèle de threads : ni *postfix*, ni *exim*, ni *qmail*, malgré leur genèse bien plus récente.

La gestion de la file d'attente des messages est un des points les plus importants d'un routeur de messages. Sous sendmail

[1] Milter – Mail Filter

8, si la gestion par défaut était suffisante lors de sa conception, elle ne l'est plus de nos jours.

Dans sa configuration par défaut, tous les messages sont mis dans une file d'attente unique, traitée périodiquement et toujours selon la même stratégie (l'âge du message, par défaut). Il est possible d'améliorer cette gestion : utilisation de plusieurs files pour créer du parallélisme, utilisation de stratégies autres que l'âge du message, traitement particulier des messages vers des destinations précises, etc. Il est aussi possible d'intégrer le résultat de la dernière connexion vers un domaine destinataire. Mais tout cela nécessite une bonne connaissance du logiciel.

Outre la stratégie d'ordonnancement, dépendante de sendmail lui-même, les opérations de lecture et écriture sur disque constituent une limitation importante dans les performances de sendmail 8. Dans les disques courants, cette limitation est de l'ordre de 120 IOPs[1], alors que la gestion de la file par sendmail, pour un message en transit, consomme de 10 à 13 opérations de lecture/écriture sur disque. On peut réduire cette limitation si l'on distribue les files de messages sur plusieurs disques ou si l'on utilise des SSD[2], dont le nombre d'IOPs peut être jusqu'à mille fois plus important que celui des disques mécaniques.

Parmi les améliorations apportées à sendmail 8, l'API milter est sûrement la plus significative. Il s'agit d'une bibliothèque permettant d'intégrer un traitement externe, en parallèle avec le traitement effectué par sendmail [7]. Pour chaque nouvelle connexion SMTP sur le serveur, le processus sendmail ouvre une connexion vers le filtre et, à chaque commande SMTP reçue par le serveur, celui-ci appelle une routine du filtre (*callback*) après avoir effectué son propre traitement. Au contraire de la plupart des autres MTAs, le traitement du filtre s'effectue en parallèle avec celui du serveur et non pas en série. Des nombreuses solutions de filtrage [8] sont apparues grâce à cette API.

Les limitations de cette API ne sont visibles que sur des serveurs de taille importante, traitant plusieurs centaines de connexions simultanées. Dans cette situation, le filtre aura à supporter autant de threads et autant de connexions ouvertes (et de descripteurs de fichier) qu'il y a des connexions ouvertes avec le serveur SMTP. Certains systèmes d'exploitation, en particulier Linux, supportent mal un processus avec un nombre important de threads. Sur sendmail 8, il est existe une implémentation alternative de la libmilter [9], utilisant le modèle « pool of workers » au lieu d'un *thread* par connexion SMTP active, ce qui permet de réduire considérablement le nombre de threads présents dans le filtre.

3. Visite guidée de sendmail X

Sendmail X essaye de rester compatible avec sendmail8 mais cette contrainte n'est pas une règle prioritaire et ne doit pas grever les choix importants de ce logiciel nouveau. Comme nous le verrons, un certain nombre de caractéristiques de sendmail 8 disparaissent avec sendmail X.

Un des défauts de sendmail 8 était son architecture monolithique – à force de vouloir satisfaire tous les besoins avec un seul module, des fonctionnalités ont été intégrées dans le MTA lui-même alors que leur place devrait être ailleurs. Un exemple est la possibilité de définition de ré-direction d'un courrier utilisant le fichier `.forward` dans le répertoire de l'utilisateur. Ceci peut-être la cause non seulement de trous de sécurité, mais aussi de baisse des performances. Pour pouvoir fournir cette fonctionnalité avec la sécurité nécessaire, il devient nécessaire d'ajouter une série de vérifications qui font encore baisser les performances. Dans de telles situations, le choix dans sendmail X est d'enlever la fonctionnalité, mais de faire en sorte que cela puisse avoir une autre solution plus simple et plus sécuritaire. Une solution à ce cas particulier est présentée dans la section *« Mise en Oeuvre »*.

3.1 Les caractéristiques de sendmail X

Le développement de sendmail X a commencé par l'établissement des caractéristiques essentielles – il s'agit de l'évident, mais qu'il fallait énumérer :

- Sécurité – il ne doit pas être possible de s'introduire dans le MTA et de compromettre la sécurité du serveur.
- Fiabilité – sendmail X ne doit perdre aucun message pour une raison « *frivole* »[10].

A ces contraintes premières, s'ajoutent d'autres objectifs de projet :

- Robustesse – le MTA doit fonctionner « raisonnablement » (mode dégradé) même en cas de défaillances. Par exemple, si le disque ou la file de messages devient plein, il doit continuer à assurer les fonctions ne nécessitant pas d'écriture sur ce disque.
- Flexibilité – Il doit être possible de remplacer ou ajouter des composants à sendmail X lorsqu'on en a besoin.
- Scabilité – sendmail X doit pouvoir profiter des améliorations apportées aux couches plus basses (système d'exploitation, configuration matérielle).
- Extensibilité – il doit être possible d'étendre sendmail X pour qu'il puisse exécuter des nouvelles fonctions ou options non prévues initialement, comme par exemple, l'enregistrement de la file de messages en attente sur un support autre que le système de fichiers UNIX classique.
- Maintenabilité – sa maintenance doit être facile, non seulement par rapport à la qualité des programmes, mais aussi en ce qui concerne la localisation d'erreurs et le comportement sans surprise des programmes.
- Portabilité – sendmail X doit fonctionner sur tous les versions des systèmes d'exploitation UNIX disponibles avec support POSIX et aussi sous Windows.
- Tests – Il est important que l'on puisse appliquer des procédures automatiques de test à la (presque) totalité des fonctionnalités de sendmail X, de façon à ce que l'on puisse prévoir, dès son installation sur une machine, le comportement final du serveur.

Mise à part la portabilité sous Windows, l'ensemble de ces pré-requis a été respecté dans le développement de sendmail X.

3.2 « Inside sendmail X »

Un des problèmes que sendmail X doit résoudre le mieux possible est la concurrence – la capacité de traitement d'un

[1] IOPs – I/O Operations per second

[2] SSD – Solid State Disk

nombre très important de connexions simultanées. En rapport avec cet objectif, un domaine de recherche ouvert est connu sous le sigle C10K[11], signifiant la capacité de traiter jusqu'à 10000 connexions simultanées. Cette référence présente efficacement les problèmes que l'on peut rencontrer.

Actuellement, si l'on veut créer des serveurs capables de traiter des débits importants avec plusieurs centaines de connexions simultanées (bien plus modestes que les 10K connexions simultanées), il est pratiquement indispensable que l'unité de traitement soit des threads et non pas des processus. La lourdeur des changements de contexte fréquents de ces derniers devient un goulot d'étranglement.

Selon la situation, sendmail X utilise un modèle de threads parmi trois : pthreads (*POSIX threads)*, *state-threads* [6] et *event-threads*.

La plupart des systèmes d'exploitation ont une implémentation de threads avec une interface de programmation compatible avec la norme POSIX [12]. Cette API offre un modèle générique assez flexible pour la plupart des applications courantes, mais elle est peu adaptée pour les applications internet telles qu'un serveur de messagerie manipulant un trafic important. Malgré une interface de programmation commune, chaque système d'exploitation implémente les threads de façon plus ou moins efficace. Les threads POSIX sont utilisés par sendmail X pour les tâches courantes n'exigeant pas des performances accrues.

Le modèle implémenté dans les « *state-threads* »[4] est essentiellement « *data-driven* ». C'est-à-dire que l'apparition d'une donnée, prête à être traitée déclenche l'apparition du nouveau thread, et non pas l'inverse, ce thread se mettra en attente d'une nouvelle donnée. Les state-threads sont surtout utilisés dans les modules client et serveur SMTP.

Le modèle implémenté par les « *event-threads* » est, comme indiqué par son nom, essentiellement événementiel et utilisé pour coordonner la plupart des tâches à l'intérieur d'un module. Il y a un superviseur en attente de nouveaux événements. Lorsqu'ils arrivent, il les décode et démarre l'action (éventuellement un pthread) pour exécuter la tâche associée à cet événement.

De ce fait le fonctionnement de sendmail X est, la plupart du temps, asynchrone. Dans un programme informatique classique, une tâche ayant besoin, par exemple, de la résolution DNS d'une adresse IP, enverra la requête au serveur DNS et restera en attente de la réponse. Sous sendmail X, le thread chargé de cette tâche envoie la requête, enregistre le contexte, et passe à une autre tâche. Lorsque la réponse arrive, le gestionnaire d'événements recherche un « *worker* » libre (ou en démarre un s'il n'y en a pas) et le désigne pour continuer le traitement de la tâche. Ainsi, les seuls threads présents sont ceux qui sont réellement actifs.

Une bonne partie de la programmation de sendmail X est organisée en couches d'abstraction, ce qui permet le remplacement aisé de modules de niveau plus bas de façon soit à suivre l'évolution des systèmes d'exploitation, soit à le porter facilement sur des systèmes autres que UNIX.

Un exemple intéressant d'abstraction concerne les routines d'accès aux données enregistrées sur disque (maps). Actuellement quatre sources d'information peuvent être utilisées :

- hash – tables d'hachage au format BerkeleyDB [13]
- passwd – listes d'utilisateurs accessibles par les routines standard du système d'exploitation
- socket – information accessible sous la forme d'un mécanisme client/serveur. Ce type de source d'information est très intéressant puisqu'il enlève la contrainte statique des bases de données classiques. Par exemple, on peut l'utiliser comme interface avec un logiciel de base de données particulier, mais on peut aussi utiliser un serveur qui retourne des informations en fonction du moment de la journée, du jour de la semaine ou qui prend en compte un paramètre quelconque.
- sequence – ce type définit une source comme étant la concaténation de plusieurs sources recherchées de façon séquentielle. Par exemple, on peut souhaiter définir la liste d'utilisateurs locaux valables comme étant ceux définis dans le fichier /etc/passwd, plus ceux présents dans une table de hachage au format BerkeleyDB.

```
map lusers { type = hash;file = "/etc/smx/localusr.db";}
map password { type = passwd;}
map valid_users {
        type = sequence;
        maps = { password, lusers };
}
```

sendmail 8 permettait la consultation de certaines sources de données externes telles LDAP. D'autres MTAs, tels *postfix* [14] et *exim* [15] intègrent directement plusieurs modèles de bases de données comme source externe d'information. Cette possibilité a été enlevée de sendmail X et, même s'il est possible de le faire, par la suite, avec l'abstraction des maps, il ne s'agit pas d'une priorité. La première raison sont les performances : s'agissant de informations en lecture seule, on peut atteindre entre 100K et 1M requêtes par seconde [16] avec des tables de hachage (Berkeley DB) – des valeurs bien supérieures à ce que l'on peut espérer avec toute autre source externe. Les autres raisons relèvent de la facilité de maintenance et de la fiabilité du MTA. Néanmoins, des solutions existent pour ceux qui ont effectivement besoin d'une source externe : soit la création périodique des tables de hachage au format BerkeleyDB à partir des autres sources (ce qui peut se faire avec des scripts assez simples) ou encore l'utilisation d'un *map* du type *socket* pour implémenter une interface en temps réel avec n'importe quel type de source de données externe.

D'autres couches d'abstraction existent, comme par exemple, les RCBs[1] permettant la communication entre modules, la CDB[2] nécessaire à l'enregistrement du contenu des messages en attente ou encore le traitement des journaux.

La nouvelle implémentation de l'API libmilter (appelée libpmilter) s'appuie sur des event-threads. Il n'y a plus un thread par connexion SMTP ouverte, mais un thread par traitement en cours sur le filtre. Aussi, la communication entre le filtre et le module smtps se fait par RCBs, multiplexée sur un seul descripteur de fichier.

[1]RCB – Record Communication Buffer

[2]CDB – Content DataBase

3.3 sendmail X – son architecture

Le choix d'une architecture modulaire a été retenu pour sendmail X dès le départ. Le point de découpage en modules essaye de répondre au mieux à des besoins de sécurité, fonctionnalité, de performance et d'extensibilité.

Les besoins de sécurité imposent que des modules touchant à des ressources différentes et non partageables ne doivent pas fonctionner sous la même identité.

Le MTA peut, par exemple, avoir un seul gestionnaire de files d'attente, alors que l'on peut vouloir avoir un serveur SMTP en écoute pour chaque interface réseau logique, avec des caractéristiques de filtrage différentes par serveur. La granularité des modules ne doit pas être trop fine car plus il y a des modules, plus les performances sont pénalisées à cause des communications inter modules. L'extensibilité doit permettre d'ajouter facilement d'autres modules.

Un MTA est essentiellement un routeur du type *« store and forward »*. Si l'on doit résumer son fonctionnement avec peu de mots, on peut dire que sa fonction est de recevoir des messages, de trouver la meilleure route vers les destinataires et les acheminer immédiatement si le destinataire est disponible ou alors de les sauvegarder pour ré-essayer plus tard.

De cette définition découlent les grandes fonctions d'un MTA : réception de messages en entrée, acheminement en sortie et gestion des messages en transit. A ces fonctions viennent s'ajouter d'autres fonctions annexes mais nécessaires. Cela correspond au choix de découpage en modules de sendmail X, qui sont le MCP, SMAR, QMGR, SMTPS et SMTPC.

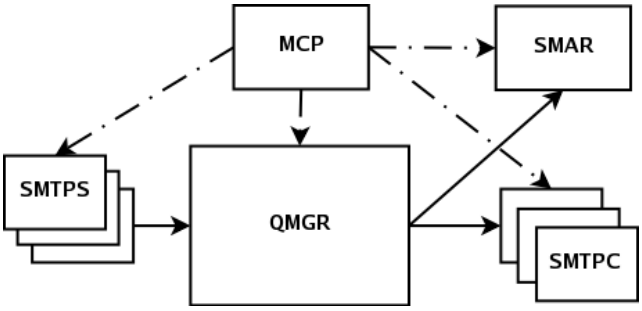

Figure 1 – *L'organisation modulaire de sendmail X*

Superviseur (MCP[1]) - C'est le module principal de sendmail X. Il est lancé au démarrage et sera chargé de démarrer tous les autres modules. En outre, il est le seul module à devoir fonctionner sous l'identité *root* : il doit pouvoir lancer un module et le faire changer d'identité et il doit pouvoir démarrer des serveurs qui écouteront sur des ports privilégiés (inférieurs à 1024 : *smtp*, par exemple). Pendant le fonctionnement de sendmail X, il doit s'assurer que chaque module continue à fonctionner et le relancer si besoin est. Pour des raisons évidentes de sécurité, il n'est pas directement accessible par un utilisateur distant.

Serveur SMTP (SMTPS[2])- c'est le module chargé de la réception des messages. Il peut y avoir plusieurs instances de ce module, qui se mettront en écoute soit sur des adresses IP différentes, soit sur des ports différents. En outre, chaque instance peut avoir ses particularités. Un exemple intéressant consiste à utiliser des instances différentes pour les messages entrants et sortants de façon à appliquer des filtres différents.

Agent de distribution (DA[3] / SMTPC[4]) – C'est le module chargé de distribuer les messages reçus après décodage de l'adresse – localement ou vers un autre serveur SMTP. Actuellement, ce module est disponible uniquement sous la forme d'un client SMTP. La distribution locale se fait par l'intermédiaire d'un LDA[5] utilisant le protocole LMTP[6] (qui est, en quelque sorte, une simplification du protocole SMTP). A l'avenir, il est possible que d'autres agents de distribution soient disponibles.

Gestionnaire de file d'attente (QMGR[7]) - Ce module est chargé de la gestion de la file d'attente des messages en sortie, et en particulier de l'ordonnancement de leur traitement. C'est un module critique, puisque de son efficacité dépendent les performances en sortie de sendmail X.

Décodeur d'adresses (SMAR[8]) - implémente la résolution d'adresses ainsi que la recherche dans les tables (map). La raison pour laquelle cette fonction correspond à un module à part est qu'il effectue des opérations dont la durée peut être longue et hors du contrôle du MTA, comme par exemple, des requêtes DNS. De ce fait, il fait aussi office de cache, de façon à ce que des multiples requêtes identiques soient traitées une seule fois. Ce module a un fonctionnement asynchrone, dans le sens où les requêtes qui lui sont faites ne sont pas bloquantes : il signale au demandeur la disponibilité de la réponse.

3.4 La gestion de la file d'attente des messages

La gestion de la file d'attente des messages en sortie est un des éléments essentiels dans l'évaluation des performances d'un MTA. Sa fonction est, d'une part d'ordonner la sortie des messages en attente de façon à ce qu'ils y restent le moins longtemps possible, tout en utilisant au mieux les ressources locales (CPU, bande passante en sortie...). Cela doit se faire avec une contrainte forte : sauf problème majeur telle la destruction du système de stockage, le MTA ne doit perdre aucun message dont il déjà a assumé la responsabilité.

Au contraire de sendmail 8, le traitement de la file de messages en sortie n'est pas périodique mais continu.

Lorsque le module SMTPS de sendmail X accepte un message il insère l'enveloppe dans la file INCEDB[9]. La INCEDB est constituée de deux copies IQDB[10] (en mémoire) et IBDB[11] (copie de sauvegarde – sur disque). Une

[1]MCP – Master Control Program

[2]SMTPS – SMTP Server

[3]DA – Delivery Agent

[4]SMTPC – SMTP Client

[5]LDA – Local Delivery Agent

[6]LMTP – Local Mail Transfer Protocol

[7]QMGR – Queue Manager

[8]SMAR – Sendmail Address Resolver

[9]INCEDB – Incoming Enveloppe DataBase

[10]IQDB – Incoming Queue DataBase

[11]IBDB - Incoming Backup DataBase

première tentative de remise est effectuée : l'enveloppe est transféré à la AQ[1].

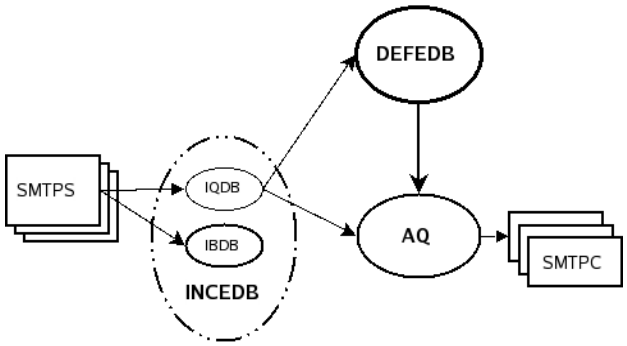

Figure 2 – *L'organisation des files de messages en attente sous sendmail X*

Sendmail X maintient les enveloppes des messages qui n'ont pas pu être remis immédiatement après leur arrivée, dans deux files logiques : la DEFEDB (file différée) et la AQ (file active), manipulées exclusivement par le module QMGR.

La file active contient les informations des messages prêts à être émis vers leur destination – ces informations sont chargées en mémoire. La file différée contient les informations sur les messages à envoyer plus tard, mais aussi des informations de la file active (si la responsabilité de la remise du message a été complètement transférée à ce MTA) – ces informations sont enregistrées sur disque.

Le traitement de ces messages est défini par deux niveaux d'ordonnancement : macro et micro. Le niveau macro est chargé de définir le moment à partir duquel une (nouvelle) tentative d'émission d'un message doit avoir lieu. A cet instant, les informations de son enveloppe sont chargées sur la AQ à partir de la DEFEDB. Le micro ordonnancement définit l'ordre dans lequel les messages de la file AQ sont émis, le nombre de connexions simultanées vers le destinataire et le nombre de messages par session. Cet ordre tient compte de la « réceptivité » de la destination. Cette « réceptivité » englobe l'ensemble de résultats précédents des envois vers cette destination : la latence, le débit obtenu, le nombre d'échecs et réussites, etc.

L'autre aspect important de la gestion de la file d'attente concerne la façon dont les messages sont stockés sur disque – ceci constitue, assez souvent, un goulot d'étranglement. Sendmail X utilise un fichier texte pour enregistrer le message et une base BerkeleyDB pour enregistrer les informations de l'enveloppe. C'est une solution qui peut sembler décevante pour ceux qui attendaient une solution où tous les messages seraient enregistrés dans un seul fichier, mais actuellement il n'y a pas de solution qui soit à la fois performante, et fiable pour assurer la contrainte de non perte de messages. Dès qu'une solution plus performante sera trouvée, elle pourra être intégrée à sendmail X, grâce à la couche d'abstraction CDB. Néanmoins, les tests obtenus en laboratoire montrent que les performances sont largement supérieures à celles obtenues avec la version 8 de sendmail.

[1]AQ – Active Queue

3.5 La configuration

sendmail X nécessite peu de fichiers de configuration. Ils se répartissent en deux catégories : le premier liste les modules à utiliser ainsi que leurs options, et les suivants particularisent le fonctionnement décrit dans le premier. Il s'agit, par exemple, des informations de contrôle d'accès, ou des informations de routage, mais toujours par rapport à des destinations spécifiques (en tant que clients ou serveurs), des adresses email spécifiques (expéditeurs ou destinataires).

Actuellement, les fichiers de configuration sont :

- /etc/smx/smx.conf – c'est le fichier principal de configuration. Ce fichier est lu initialement par MCP, lui permettant d'identifier les modules à démarrer et avec quelles options, puis par chaque module, leur permettant de définir ses options de fonctionnement.
- /etc/smx/aliases.db – C'est le fichier des aliases, comme existait déjà dans sendmail 8, mais qui maintenant inclut aussi les destinataires des domaines virtuels.
- /etc/smx/access.db – C'est une table dont le contenu est semblable à celui de la table de même nom sous sendmail 8 : définir des actions/contraintes d'accès en fonction de l'adresse ou nom du client SMTP, de l'adresse e-mail de l'expéditeur ou du destinataire. Ces sont des informations intrinsèquement utiles au module SMTPS.
- /etc/smx/mt.db – mailertable ou table de routage. C'est sur ce fichier que l'on va définir vers où sendmail doit envoyer les messages en fonction du domaine. Ce fichier qui n'était pas très important sur sendmail 8, devient essentiel, puisque c'est dans ce fichier que l'on définira les domaines locaux et les domaines virtuels.
- /etc/smx/qmgr_conf.db – c'est le fichier de paramètres de configuration de QMGR. Il contient des paramètres spécifiques à chaque adresse IP, soit client soit serveur SMTP, comme, p.ex., le nombre maximal de connexions simultanées à ouvrir vers une destination donnée.

Ces fichiers constituent l'essentiel. La très grosse majorité des situations peut être résolue avec les fichiers ci-dessus. Sauf réel besoin, la possibilité de faire la même chose à plusieurs endroits a été réduite au maximum.

Le fichier de configuration principal est structuré en sections. L'exemple ci-après montre la structure d'un fichier de configuration minimal et seule la section concernant le serveur SMTP est détaillée :

```
lmtp       { ... }
smtps MTA {
   log_level = 11;
   log { facility = mail; ident = "MTA"; }
   CDB_gid = 262;
   wait_for_server = 4;
   listen_socket { type = inet; port = 25; }
      start_action = pass;
   pass_fd_socket = smtps/mtaintfd;
   user = smxs;
   path = "/opt/smx/libexec/smtps";
   arguments = "smtps -I 1 -N MTA -f /etc/smx/smx.conf";
   flags = { access };
   pmilter {
      socket { type = inet;
        port = 2200; address = 127.0.0.1;}
   }
}
```

```
smtpc    { ... }
qmgr     { ... }
smar     { ... }
```

D'autres exemples détaillés sont disponibles sur le site sendmail-fr [17].

Un serveur de messagerie typique dont la configuration est proche de celle par défaut nécessite encore la présence des sections définissant les modules. Dans ces conditions, le fichier de configuration principal contient moins d'une centaine de lignes.

Tous les modules acceptent des personnalisations qui peuvent soit concerner l'activation d'options qui ne le sont pas par défaut, ou alors des optimisations telles le changement de certains délais d'attente nécessaires à un serveur de listes de taille importante. Pour avoir une idée de ce que cela peut représenter, la section *qmgr* ne contient que 12 lignes dans sa configuration minimale, mais l'inclusion de toutes les options possibles la fait monter à 174 lignes.

Pour obtenir la configuration par défaut d'un module, il suffit de le lancer avec l'option «-*VVVVV*» ou «-*VVVVVV*». Par exemple :

```
/opt/smx/libexec/qmgr -VVVVVV
```

4. Sendmail X – mise en oeuvre

Sendmail X étant un logiciel complètement différent de sa version précédente, il n'est pas possible de mettre à jour une installation à partir d'une installation de sendmail 8 déjà existante.

Le fichier principal de configuration smx.conf est à créer complètement. Certains autres fichiers (aliases, par exemple) peuvent être créés à partir des fichiers déjà existants.

La documentation README[18] présente dans la distribution de sendmail X est la meilleure référence pour sa mise en oeuvre. Le site sendmail-fr [17] présente des conseils et des outils destinés à la mise en place de sendmail X, et en particulier, les situations ci-après.

Le but de ces exemples n'est pas de présenter exhaustivement la mise en oeuvre de sendmail X, mais de montrer brièvement quelques nouveautés.

4.1 L'existence des utilisateurs locaux

Au contraire de la version précédente, sendmail X ne consulte pas, par défaut, la base des utilisateurs locaux pour vérifier leur existence. En effet, en tant que MTA, la seule information dont il a besoin est l'existence ou pas d'un utilisateur local. Il serait même déconseillé, du point de vue de la sécurité, de permettre l'accès, par le MTA, aux mots de passe des utilisateurs, même chiffrés.

Le mode de vérification de l'existence des utilisateurs locaux doit être déclarée explicitement :

- déclaration de tous les utilisateurs locaux dans le map aliases. Par exemple :

```
joe      :      local:
```

- utilisation d'une map externe et spécification dans la section SMAR (avec ré-utilisation de la table d'utilisateurs donnée comme exemple précédement :

```
map lusers {
        type = hash;
        file = "/etc/smx/localusr.db";
}
map password { type = passwd; }
map valid_users {
        type = sequence;
        maps = { password, lusers };
}
local_user_map {
        name = valid_users;
        flags ={ implicitly_match_detail };
}
```

Cette deuxième forme est bien plus pratique et permet l'utilisation simultanée de deux sources de données. C'est de cette façon que l'on vérifiera l'existence des utilisateurs locaux sous Cyrus IMAP.

4.2 Définition des domaines locaux ou virtuels et de ses utilisateurs

Sous sendmail 8, les domaines locaux étaient définis dans le fichier `/etc/mail/local-host-names` (classe w) et les domaines virtuels soit dans ce même fichier, soit dans la table de routage (*mailertable*) ou encore avec l'aide de la macro `VIRTUSER_DOMAIN` (classe VirtHost), Sous sendmail X, ces déclarations se font toutes dans la table de routage (`/etc/smx/mt`) :

```
undomaine.com          lmtp:
autredomaine.com       2002 ^ lmtp:localhost
```

Dans le premier exemple, la remise du message se fait utilisant le protocole `lmtp` et la socket définie dans la section SMTPC du fichier de configuration. Dans le deuxième cas, il s'agit aussi du protocole `lmtp`, mais avec un autre LDA. Dans ces cas, domaine.com et autredomaine.com sont des domaines virtuels, mais dont le type de gestion de boîtes aux lettres peut être différent – par exemple : boîte aux lettres classique pour le premier et IMAP pour le deuxième.

Le deuxième point consiste à définir les utilisateurs valides pour ces domaines. Dans la version 8, il fallait utiliser deux tables : `aliases`, pour les adresses sans la partie domaine et `virtusertable` pour les adresses avec la partie domaine. La table des `aliases`, acceptant maintenant des adresses avec partie domaine, est devenue suffisante.

4.3 Utilisation de Cyrus IMAP comme LDA

L'utilisation de Cyrus IMAP [19] avec sendmail 8 présentait un grand inconvénient : sauf modifications peu conventionnelles, il était impossible de vérifier l'existence d'une boîte aux lettres avant que le message soit transmis au LDA. Avec sendmail X, il suffit de lister les boîtes aux lettres IMAP existantes et créer une map avec cette liste.

4.4 Soumission de messages en ligne de commande

La version actuelle de sendmail X n'est pas encore livrée avec un MSP[1].

Si le besoin est simple, il est possible d'utiliser mini_sendmail [20] ou nbSMTP [21]. Ce sont des outils assez simples mais efficaces permettant de reproduire la syntaxe de sendmail 8 en ligne de commande pour l'envoi de messages. Néanmoins, ces outils ne permettent pas de

[1]MSP – Message Submission Program

différencier les erreurs selon leur type et origine, ni de gérer une file, si le serveur d'envoi n'est pas présent.

Une deuxième solution, plus complexe, mais beaucoup plus sécurisée est d'utiliser la partie soumission de sendmail 8. Dans ce cas, le seul fichier de configuration a récupérer de sendmail 8 est celui permettant la soumission de messages (submit.cf). Les administrateurs qui, pour un souci de fiabilité de migration, ne souhaitent pas utiliser l'installation de sendmail 8 existante pour la soumission de messages, peuvent installer sendmail 8 dans une arborescence à part.

4.5 La ré-direction et les fichiers « *.forward* ».

sendmail 8, ainsi que la plupart des MTAs, vérifie l'existence d'un fichier de nom « .forward » dans le répertoire home de l'utilisateur. Ce fichier sert à re-diriger les messages d'un utilisateur vers une autre adresse. Sendmail X ne le fait plus pour des raisons à la fois de performance, de fiabilité, mais aussi puisqu'il s'agit d'un point d'entrée pour des problèmes de sécurité. Mais, en réalité, cela ne constitue une perte considérable. Actuellement, dans la plupart des organisations, les utilisateurs n'ont plus accès au serveur de messagerie. Pour des raisons de fiabilité, les répertoires des utilisateurs ne sont plus montés par le serveur de messagerie.

Une solution de remplacement consiste à utiliser un script qui parcourt les répertoires des utilisateurs à la recherche des fichiers de re-direction, qui collecte leur contenu et qui crée un fichier avec les entrées correspondantes qui seront ajoutées à la table des `aliases` de sendmail X. Le résultat de ce traitement est alors transféré périodiquement vers le serveur de messagerie. Une autre alternative consiste à utiliser des formulaires web permettant de gérer ces fichiers dans un répertoire unique du serveur de messagerie. Cette dernière solution est déjà utilisée couramment dans les organisations où les utilisateurs possèdent un compte de messagerie, mais pas de compte avec un interpréteur de commandes.

4.6 Utilisation d'un serveur de stockage autre que le serveur SMTP

Une situation que l'on retrouve de plus en plus souvent est la répartition de la charge de la messagerie entre deux machines, la première faisant office de serveur de réception et la deuxième remplissant les fonctions de stockage des messages et de serveur POP ou IMAP. L'inconvénient de la version précédente de sendmail X est que le transfert des messages entre les deux machines nécessitait, la plupart du temps, un deuxième serveur SMTP fonctionnant sur le serveur de stockage.

Avec sendmail X, il suffit de mettre sur la deuxième machine un serveur LMTP en écoute sur un port TCP et configurer sendmail X pour envoyer les messages vers les destinataires locaux directement au serveur de stockage. Il suffit de mettre la ligne suivante dans le fichier /etc/smx/mt :

```
domaine.local        2003 ^ lmtp:imap.mondomain.com
```

4.7 Des adresses protégées

Un des souhaits courants est la protection de certaines adresses. Par exemple, on souhaite créer un alias (`tous_ecole`) permettant d'atteindre tous les utilisateurs de l'organisation. Typiquement, on souhaitera limiter l'accès à cette adresse uniquement aux utilisateurs dans le réseau local ou alors à ceux qui appartiennent à la liste des destinataires. Cet exemple est, le plus souvent résolu avec un gestionnaire de listes de diffusion (Sympa[22], par exemple), mais cette solution devient trop lourde pour des adresses de service ou alors lorsque la protection est la seule utilité du serveur de listes.

Un exemple de mise en place serait :

- ajouter une sous-section dans la section SMTPS :

```
protected_recipients {
        allow_by { sender, client_IP};
}
```

- Définir des adresses dans la table des `aliases` :

```
list1:    <usr1@d1.dom> <usr2@d2.dom>
list2:    <usr3@d3.dom> <usr4@d4.dom>
```

- Définir les protections dans la table `access`

```
protectedrcpt:list1@local.dom        list:<list1@local.dom>
protectedrcpt:list2          from:<joe@local.dom> cltaddr:10
```

Dans cet exemple, seuls les abonnés de list1@local.dom peuvent envoyer des messages à cet adresse, tandis l'accès à list2 est limité à joe@local.dom ou aux clients SMTP du réseau privé 10/8.

5. sendmail X – mesures de performance

On trouve dans la documentation de projet [16] les résultats des expérimentations aussi bien de sendmail X, mais aussi de certaines de ses parties. Celle qui semble être la plus représentative dans le cas d'un serveur de messagerie est sa capacité de relayage.

Des essais comparatifs avec sendmail 8.13.4 sur un serveur Sun E450 ont été réalisés par Claus Assmann [16] et [23].

Il s'agit d'injecter un nombre important de messages à l'entrée du MTA et de mesurer le temps nécessaire pour que tous les messages soient sortis de la file de messages. Les mesures ont été faites pour ne prendre en compte que les facteurs intrinsèques au MTA, toute influence externe devant être éliminée, en particulier, les requêtes DNS.

Deux outils auxiliaires[1] sont utilisés :

- `smtpc`, un client SMTP, source des messages à relayer, selon les paramètres passés en ligne de commande.
- `smtps`, un serveur SMTP fonctionnant comme puits. Après son démarrage, il attend un nombre de messages (passé comme paramètre en ligne de commande) et à la fin, il présente l'intervalle de temps écoulé entre les messages extrêmes. On déduira la capacité de relayage, comme étant le ratio entre le nombre de messages reçus et le temps écoulé.

Le MTA en cours de test est configuré pour envoyer tous les messages en sortie vers le puits.

L'outil `smtpc` est démarré pour envoyer 100000 messages de taille 4000 octets en 10000 sessions (c'est-à-dire, 10 messages par session). Il doit ouvrir 100 sessions simultanées avec le MTA.

[1]Ces outils de test sont fournis avec la distribution de sendmail X et se trouvent dans le répertoire *statethreads/examples*

La machine de test est une Sun Enterprise E450 sous Solaris, avec 4 processeurs tournant à 400 Mhz.

Sendmail X Beta.0.0.0	***Temps***	***Débit***
Conf ig par défaut	924 s	108,2 msgs/sec
Sans fsync	585 s	170,2 msgs/sec
Utilisation de tmpfs	542 s	184,2 msgs/sec

Table 1 – *Mesure du temps nécessaire à sendmail X pour relayer 100000 messages de 4000 caractères avec 100 sessions parallèles*

Dans la configuration par défaut, lorsque le module SMTPS accepte un message, son écriture sur disque est suivie d'une opération COMMIT (en réalité, un appel à f`sync(2)`) avant de signaler son acceptation au client SMTP – il n'y a donc pas de risque de perte de message.

Sans cette option, le choix de la date d'écriture effective sur disque sera une décision du système d'exploitation et, par conséquent, pas forcément immédiate. Les performances obtenues dans cette situation sont meilleures, mais le risque de perte de messages n'est pas nul en cas de faille système ou d'arrêt inopiné du serveur.

Dans la troisième situation, les messages en attente sont enregistrés en mémoire virtuelle (réelle ou swap) : dans cette situation, un arrêt ou redémarrage de la machine implique la perte complète des messages en attente.

Les mêmes essais ont été réalisés avec sendmail 8 sur la même machine. Pour que les résultats soient comparables, et pour obtenir des meilleures performances avec sendmail 8, il a été configuré avec 16 files d'attente en sortie.

Sendmail 8.13.4	***Temps***	***Débit***
DeliveryMode=background	2307 s	43,3 msgs/s
DeliveryMode =interactif	860 s	116,3 msgs/s
DeliveryMode=interactif, Supersafe=false	668 s	149,7 msgs/s

Table 2 – *Mesure du temps nécessaire à sendmail 8 pour relayer 100000 messages de 4000 caractères avec 100 sessions parallèles*

En ce qui concerne sendmail 8, seule la première situation d'essai est vraiment comparable avec la première situation de sendmail X. Dans la troisième situation il n'y a pas d'enregistrement des messages sur disque, tandis que dans la deuxième, seuls les messages qui ne peuvent pas être relayés immédiatement sont enregistrés.

Très peu d'études comparatives des performances entre les différents MTAs sont disponibles. L'étude de Mathias Andree [23] est celle semblant être la plus pertinente, mais son but principal était surtout la comparaison entre *qmail* et *postfix*. Des informations sur sendmail ont été ajoutées, mais probablement dans un but accessoire, on peut donc s'interroger sur leur pertinence. Dans la plupart des mesures, la capacité de relayage obtenue avec *postfix* a été de l'ordre de une fois et demie celle de *sendmail 8,* la capacité de ce dernier étant meilleure, mais comparable à celle de *qmail*.

Ces mesures sont intéressantes, et on serait tenté de les utiliser ensemble pour comparer, même grossièrement, sendmail X avec d'autres MTAs. Même s'il serait hasardeux d'avancer des chiffres, on peut avoir un ordre d'idée si l'on tient compte des restrictions suivantes :

- La version de sendmail (8.11.6) utilisée par Matthias Andree est largement dépassée (et l'était déjà lors des essais). La version courante (8.13.4) est plus rapide.
- sendmail 8.11.6 a été configuré pour utiliser une seule file d'attente, alors que dans les experimentations effectuées par Claus Assmann, il y en avait 16, traitées en parallèle et améliorant significativement les performances de sendmail 8.
- Le nombre de messages émis est une restriction importante. Utiliser un nombre bien plus petit de messages à relayer permet d'évaluer le comportement vis à vis de rafales courtes, mais pas vis à vis d'une charge soutenue.
- Le petit nombre de connections simultanées peut aussi devenir trompeur, puisqu'il ne permet pas d'évaluer la capacité de traitement parallèle du MTA. Certains MTAs peuvent s'écrouler rapidement avec un nombre important de connexions simultanées, soit à cause du MTA lui-même, soit à cause des caractéristiques du système d'exploitation (Linux, par exemple, à cause de son implémentation des threads POSIX).

Ces observations sont résumées dans le tableau ci-après :

	Benchmark smX-sm8 C. Assmann	***Benchmark postfix-qmail-sm8 M. Andree***
Ordinateur	Sun E450 SunOS 4x400MHz	X86 FreeBSD – 4.4 300MHz
Version de sm8	8.13.4	8.11.6
Nombre de files	16	1
Messages	100000	1000
Connexions simultanées	100	5
Taille des messages	4000	3000
Débit sm8	43,2 msgs/s	7 – 10 msgs/s

Table 3 – *Conditions de mesure dans les expérimentations menées par Claus Assmann et par Matthias Andree*

6. sendmail X – état du développement

Au moment de l'écriture de ce papier, la version courante de sendmail X était la 0.0.Beta-4.0. Cela veut dire que les fonctionnalités de la première version sont figées et que, sauf erreurs ou besoins mineurs ne mettant pas en cause de façon substantielle les fonctionnalités actuelles, il n'y aura plus de changements avant la première version officielle : sendmail X 0.0.0.

Certaines fonctionnalités que l'on trouve dans d'autres MTAs sont manquantes, mais la sortie d'une version dans

l'état actuel se justifie par le besoin de valider et d'obtenir des résultats sur ce qui a déjà été fait par une communauté plus large que celle qui a spécifié et effectué des tests préliminaires, constituée en majorité des experts.

Deux catégories de fonctionnalités sont manquantes : des modules à ajouter et des modifications dans les modules déjà existants.

- MSP – Dans la version actuelle, la soumission de messages en ligne de commande nécessite un programme non fourni avec sendmail X. La meilleure solution actuellement disponible est la version intégrée dans sendmail 8.
- LDA – Le besoin d'un LDA se présente dans trois situations : la remise des messages dans la boîte aux lettres de l'utilisateur, son enregistrement dans un fichier particulier ou alors l'envoi par un tube (pipe) vers un programme quelconque. Les deux dernières situations étaient traitées en interne par sendmail 8, sans faire appel à un programme extérieur : ce n'est plus le cas, et ce ne sera probablement pas, même dans une version future, puisqu'il s'agit de faire faire par le MTA une tâche qui doit être remplie par un module externe. Ces besoins peuvent être remplis par *procmail*[24] (LDA conseillé actuellement comme agent de remise local pour les boîtes aux lettres au format mbox), mais dont la mise en oeuvre est laborieuse à cause de la complexité de son langage de configuration. Il est, donc possible que sendmail X aura un jour un module LDA, comme c'était déjà le cas pour sendmail 8.
- Un filtre milter ne peut pas modifier le contenu de la partie DATA d'un message. Il n'est donc pas possible ni de remplacer ni de modifier le corps du message, ni d'ajouter ou supprimer des en-têtes. C'est en fait la restriction la plus gênante, puisqu'elle empêche toute communication entre un filtre et le logiciel client MUA[1].
- API milter compatible avec celle de sendmail 8. Mise à part la restriction du paragraphe précédent, l'utilisation avec sendmail X des filtres écrits pour sendmail 8 exige la ré-écriture des parties du filtre faisant appel aux fonctions de la *libmilter*. Une version future de sendmail X proposera une couche logicielle de compatibilité avec la bibliothèque milter de sendmail 8.
- Les options de ré-écriture des adresses, telles que proposées par sendmail 8 ne sont pas disponibles (*masquerade*, *genericstable* et *domaintable*). Elles le seront dans le module de soumission de messages. Ces options ne sont pas bloquantes, puisqu'elles peuvent être configurées sur les clients de messagerie.
- Il manque la possibilité de gestion d'adresses de bounce (par exemple, VERP[2]), nécessaire aux serveurs de liste de diffusion (Sympa, par exemple). Sans cela, le serveur de listes doit émettre un message par destinataire, alors que si cette gestion est faite par le MTA, il y a des gains dans l'émission des messages par le serveur de listes, dans la gestion de la file par le MTA et dans les performances de l'ensemble.

[1]MUA – Mail User Agent

[2]VERP -

7. Conclusion

Après avoir servi pendant plus de dix ans, la nouvelle version de sendmail est une révolution par rapport à la version précédente. Cette version a été complètement ré-écrite pour pouvoir tirer parti des caractéristiques des systèmes d'exploitation modernes.

Elle est sûrement en avance par rapport à ses concurrents actuels, aussi bien dans la partie apparente (les performances) que dans sa partie cachée (ses possibilités d'évolution, flexibilité...).

Les anciens utilisateurs de sendmail trouveront des situations familières, mais constateront aussi la disparition d'autres fonctionnalités. Ces disparitions peuvent paraître gênantes, à première vue, mais l'auteur n'a pas encore trouvé de situation insurmontable et, globalement, les gains compensent largement ce qu'on pense avoir perdu.

Il est maintenant à souhaiter que cette nouvelle version de sendmail rencontrera tout le succès qu'elle mérite.

8. Remerciements

Je tiens à remercier Claus Assmann, d'abord par le développement de sendmail X, outil qui, je l'espère, sera utilisé autant ou plus que les versions précédentes de sendmail. Je le remercie aussi par la patience avec laquelle il a répondu à toutes mes questions et m'a aidé à trouver des solutions aux problèmes rencontrés.

D'autres amis ont relu ce papier, commenté et proposé des idées et des changements. Je les remercie anonymement – ils se reconnaitront.

Bibliographie

[1] B. Costales et E. Allman, *sendmail*. O'Reilly 3rd edition, Décembre 2002

[2] *Le kit de configuration de sendmail de Jussieu* – http://www.kit-jussieu.org

[3] D. J. Bernstein, *Mail Disasters.* http://cr.yp.to/maildisasters.html

[4] *Sendmail Coding Standards* – http://www.sendmail.org/CodingStardard.html

[5] T. Miller et Theo de Raadt, *strlcpy and strlcat - Consistent, Safe, String Copy and Concatenation*, 1999 USENIX Annual Technical Conference - http://www.usenix.org/events/usenix99/millert.html

[6] *State Threads for Internet Applications*, http://state-threads.sourceforge.net/docs/st.html

[7] B. Costales et M. Flint, *Sendmail Milters,* Addison-Wesley, 2005

[8] Stephane Lentz, *Milters, Introduction and Product List* - http://milter.free.fr/intro/index.html

[9] J. Martins da Cruz, *libmilter with a pool of workers,* http://j-chkmail.ensmp.fr/libmilter/

[10] J. Klensin, RFC 2821 – *Simple Mail Transfert Protocol*,

[11] D. Kegel, *The C10K Problem, http://www.kegel.com/c10k.html*

[12] IEEE, *IEEE/ANSI Std 1003.1 – Information Technologie – Portable Operating System Interface (POSIX)*, 1996, IEEE Standards Press

[13] *Berkeley DB database*, http://www.sleepycat.com

[14] W. Venema, *Postfix Documentation*, http://www.postfix.org/documentation.html

[15] P. Hazel, *Exim, http://www.exim.org*

[16] C. Assmann, *sendmail X : Requirements, Architecture, Functional Specification, Implementation and Performance*, May 2005, Emeryville CA

[17] *Non-official ressources for sendmail*, http://www.sendmail-fr.org/smx

[18] C. Assmann, *sendmail X – README,* Sep 2005, Emeryville

[19] *Cyrus IMAP Server*, http://asg.web.cmu.edu/cyrus/

[20] *mini_sendmail* , http://www.acme.com/software

[21] *no-brainer SMTP*, http://nbsmtp.ferdyx.org

[22] *Sympa*, http://www.sympa.org

[23] http://www-dt.e-technik.uni-dortmund.de/~ma/postfix/vsqmail.html

[24] procmail web page, http://www.procmail.org