Gestion de réseaux IP basée sur les technologies Web

IMAG, Grenoble, France
11 février 1999

Jean-Philippe Martin-Flatin
École Polytechnique Fédérale de Lausanne
Institut pour les Communications informatiques et leurs Applications
ICA
martin-flatin@epfl.ch
http://icawww.epfl.ch/~jpmf/

Plan

- Plateformes de gestion de réseaux pré-Web
- Problèmes
- Gestion de réseaux IP basée sur les technologies Web
- Modèle *pull*
- Modèle *push*
- Recherche actuelle et future
- Q&R

Plateformes de gestion de réseaux pré-Web

- Télécoms :
  - ITU-T Rec. X.700 : 5 domaines de gestion de systèmes (FCAPS)
- IP :
  - 3 fonctions principales (obligatoires) :
    - monitoring du réseau
    - collecte de données
    - traitement des notifications
  - autres fonctions (optionnelles) : inventaire, configuration, contrôle d'accès, facturation...
  - 3 cadres (*frameworks*) de gestion SNMP : SNMPv1, SNMPv2c et SNMPv3
  - Exemples de PGR : HP OpenView, Cabletron Spectrum, IBM Netview, Sun Solstice...

PGR : 3 fonctions principales

- Monitoring (court terme) :
  - détecter les fautes survenant dans les équipements et liens réseaux :
    - ➠ réactif par rapport aux fautes
    - ➠ préventif par rapport aux plaintes des utilisateurs ou clients
- Collecte de données (long terme) :
  - rassembler des données statistiques pour constituer les rapports d'utilisation quotidiens, hebdomadaires et mensuels :
    - ➠ préventif par rapport aux pannes à long terme
- Traitement des notifications (très court terme) :
  - réactif par rapport aux fautes
  - réagir à des évènements générés par les agents (notifications SNMP)
  - réagir à des évènements générés par le gestionnaire (corrélateur d'évènements)

Gestion régulière

- Monitoring et collecte de données
- Mode continu, sans interruption
- Automatisation = 2 modes :
  - avec présence humaine : opérateurs devant des interfaces graphiques 24h/24 (syndrome de l'icône rouge)
  - sans présence humaine : déclenchement automatique d'alarmes (*pager*, bip, courrier électronique, téléphone, sirène...)
- Types de réseaux: grands ou moyens

Gestion *ad hoc*

- Réparation ou configuration
- Mode manuel
- Présence humaine nécessaire : administrateur ou opérateur
- Types de réseaux: grands, moyens ou petits
- Petits réseaux :
  - la gestion *ad hoc* remplace la gestion régulière
  - pas de gestion préventive

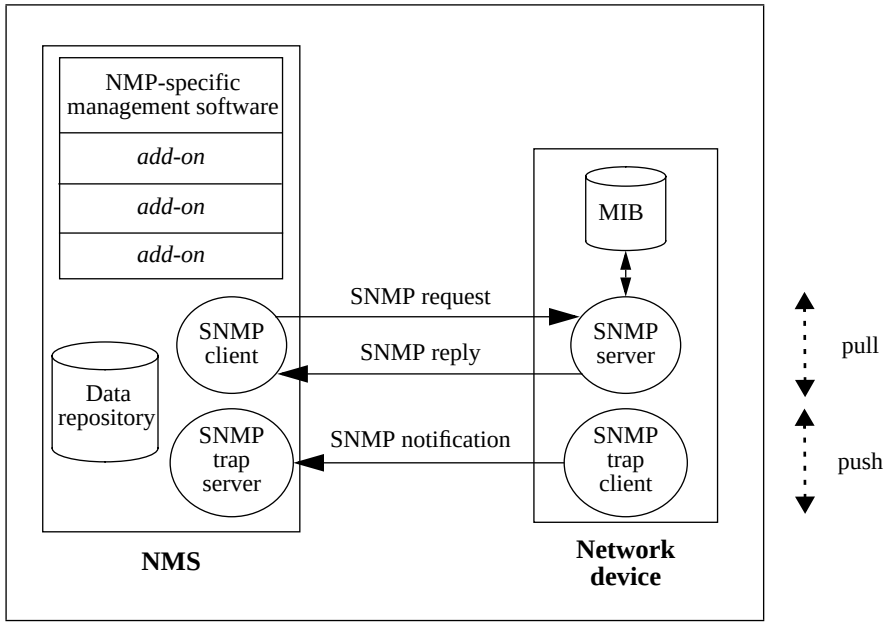
PGR pré-Web: cadre de gestion SNMP

- Paradigme gestionnaire/agent
- *Polling* pour le monitoring et la collecte de données
- *Push* non sollicité pour l'envoi des notifications
- Protocole de communication SNMP
- MIB génériques (par ex. MIB-II)
- MIB propriétaires (par ex. Cisco)
- SMIv2 (dérivé de ASN.1)
- Encodage BER
- ...

PGR pré-Web : Évolution du marché

- Il était une fois les systèmes ouverts... :
  - équipements réseaux génériques
  - gestion générique
- Segmentation du marché [JPMF SSC/1998/021]
- MIB génériques --> MIB propriétaires
- Interfaces graphiques de gestion génériques --> propriétaires (*add-ons*, par ex. CiscoWorks)
- Pour les clients, le caractère "ouvert" de la gestion dans le monde IP n'est plus garanti que par :
  - le cadre de gestion SNMP
  - le protocole de communication SNMP

Modèle simple de PGR pré-Web



Problèmes avec les PGR pré-Web (1/2)

- Clients :
  - les PGR coûtent trop cher (matériel et logiciel) :
    - ➠ matériel dédié à la gestion de réseaux
  - le support de BD relationnelles tierce-parties est trop limité
    - ➠ le client dépend d'accords bilatéraux entre vendeurs de PGR et de BD rel.
  - besoin d'expertise Unix pour la maintenance des PGR existants :
    - ➠ coût de migration vers Windows trop élevé
- Vendeurs de matériel :
  - coûts de développement des *add-ons* propriétaires trop élevés :
    - ➠ nombreux équipements
    - ➠ nombreux PGR
    - ➠ nombreux systèmes d'exploitation

Problèmes avec les PGR pré-Web (2/2)

- Clients et vendeurs de matériel :
  - temps de mise sur marché des *add-ons* trop long :
    - ➠ plusieurs mois après la mise en vente du matériel si le vendeur a une part de marché importante
    - ➠ jamais si le vendeur a une faible part de marché :
      - les *startups* ont besoin d'un PGR séparé
  - gestion de versions multiples d'une MIB :
    - ➠ clash entre la version supportée par l'*add-on* du PGR et celles supportées par les différents agents ; on doit :
      - soit mettre à jour le PGR manuellement, agent par agent
      - soit ne pas utiliser les nouvelles fonctionnalités de la MIB tant que tous les agents n'ont pas été mis à jour

Problèmes d'ingénierie avec la gestion de réseaux pré-Web (1/4)

- Efficacité du protocole :
  - inefficacité de l'encodage BER [Mitra 1994] [Neufeld and Vuong 1992] :
    - ➠ amélioré par l'encodage PER dans le modèle OSI
    - ➠ SMIv1 et SMIv2 : BER obligatoire pour tous les cadres de gestion SNMP
  - inefficacité de SNMP :
    - ➠ pas de moyen efficace de récupérer une table entière :
      - nombreux échanges de messages
      - effet négatif sur la latence et le *network overhead*
    - ➠ dans une *varbind list*, la description des OID prend beaucoup plus de place que les valeurs de ces OID

Problèmes d'ingénierie avec la gestion de réseaux pré-Web (2/4)

- Sécurité :
  - pas d'opération `get` ou `set` sécurisée dans SNMPv1 et SNMPv2c
  - SNMPv3 :
    - ➠ récent
    - ➠ n'a pas encore fait ses preuves en industrie
  - gestion à distance d'une filiale (VPN) : matériel coûteux pour coder/décoder de façon transparente aux frontières du réseau
  - *firewalls* : relais UDP non triviaux à configurer et à maintenir [Chapman and Zwicky 1995]
    - ➠ problème pour les PME

Problèmes d'ingénierie avec la gestion de réseaux pré-Web (3/4)

- Protocole de transport :
  - en théorie : SNMP accepte TCP comme UDP
  - en pratique : seul UDP est utilisé
  - pas d'accusé de réception :
    - ➠ perte de notifications SNMP importantes pour des raisons futiles, par ex. débordement d'une file d'attente dans un routeur IP (*buffer overflow*)
    - ➠ résolu dans SNMPv3 avec `inform`
  - certaines données de gestion sont plus critiques que des données "utiles" utilisateurs

Problèmes d'ingénierie avec la gestion de réseaux pré-Web (4/4)

- Sémantique :
  - modèle conceptuel de l'application de gestion de réseaux limité par une sémantique de bas niveau :
    - variables de MIB
    - opérations `get`, `getnext`, `set`...

Gestion basée sur les technologies Web: définition

- Réponse marketing = WBEM
- Réponse technique = utilisation des technologies du Web pour faciliter la gestion intégrée des réseaux, des systèmes et des services :
  - formulaires HTML
  - applettes, servelettes ou applications Java
  - JDBC
  - Java RMI
  - sérialisation d'objets Java
  - Java IDL (CORBA)
  - ...

Gestion basée sur les technologies Web: distribution

- Communication au sein d'une application distribuée :
  - HTTP
  - *sockets* (TCP ou UDP)
  - Java RMI
  - Java IDL (CORBA) --> télécoms
- En général :
  - entre applette et servelette Java : HTTP ou *sockets*
  - entre applette et application Java : *sockets* ou Java RMI

Que peuvent apporter les technologies Web ?

- Hier : HTML
- Aujourd'hui : modèle *pull*
- Demain : modèle *push*
- Après-demain: code mobile (tâches de gestion déléguées aux agents)

Pages HTML

- Tâches secondaires:
  - automatisation et standardisation de la remontée et de la saisie des problèmes réseaux (*helpdesk*)
  - mise en ligne des rapports d'utilisation du réseau (politique du "bureau sans papier")
  - aide en ligne pour la réparation des réseaux (*troubleshooting*) :
    - ➠ les administrateurs écrivent des pages HTML de type symptôme-action pour les opérateurs, avec des hyperliens vers les documentations en ligne des vendeurs
    - ➠ accès simple aux scripts (Perl, Tcl/Tk) et aux programmes (`ping`, `traceroute`, `netstat`) de gestion de réseaux

Interface de commandes HTML

- Correspondance simple entre URL et interface de commandes :
  - par ex. sur les routeurs Cisco [Bruins 1996] :
    - `http://routername/exec/show/interface/ethernet0/`
    - `show interface ethernet0`
  - URL générées dynamiquement depuis des formulaires HTML
  - URL insérées comme hyperliens dans des pages HTML de type symptôme-action

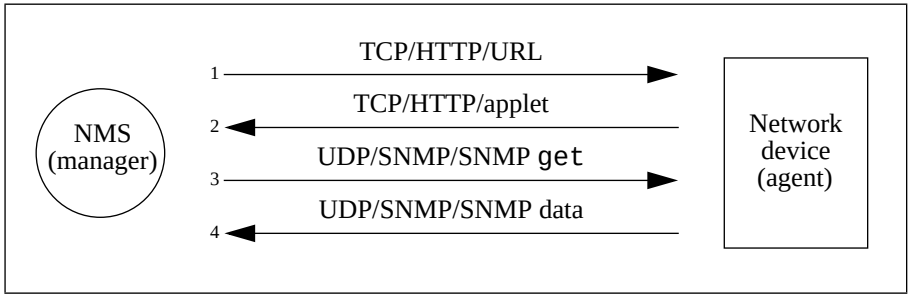
Les modèles *pull* et *push*

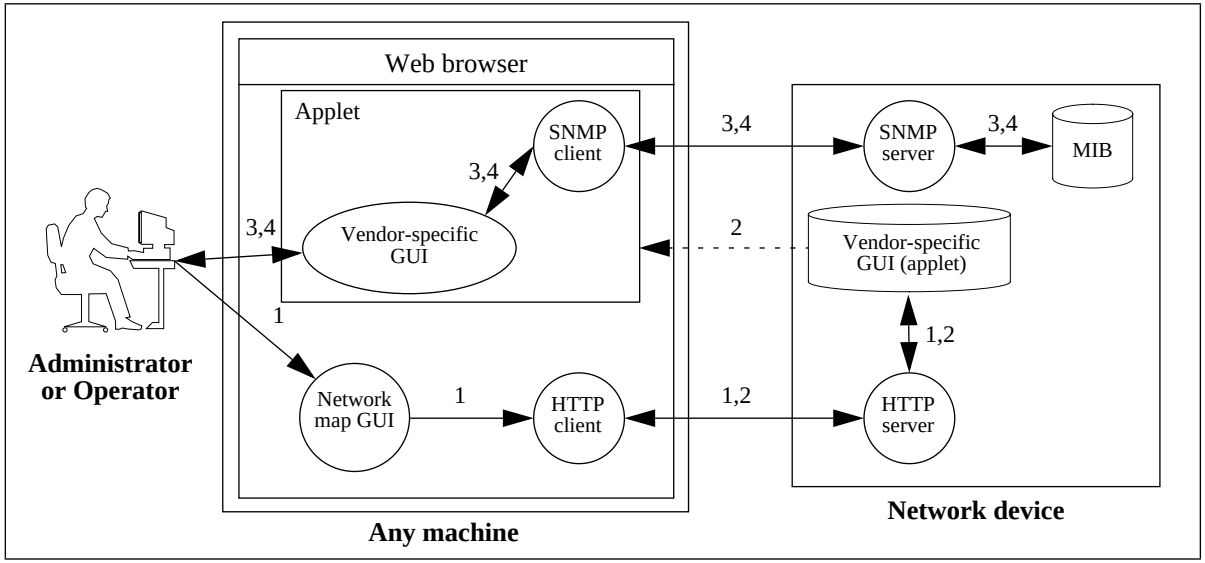
- Métaphore du journal :
  - soit on l'achète tous les jours chez le marchand de journaux (*pull*)
  - soit on le reçoit tous les jours par la poste (*push*)
- Modèle *pull* :
  - paradigme requête-réponse
  - transfert de données initié par le gestionnaire
  - ex. : *polling* dans les PGR pré-Web (monitoring et collecte de données)
- Modèle *push* :
  - paradigme publication/souscription
  - transferts de données parallèles et indépendants initiés par les agents
  - ex. : notifications SNMP dans les PGR pré-Web

Modèle *pull*

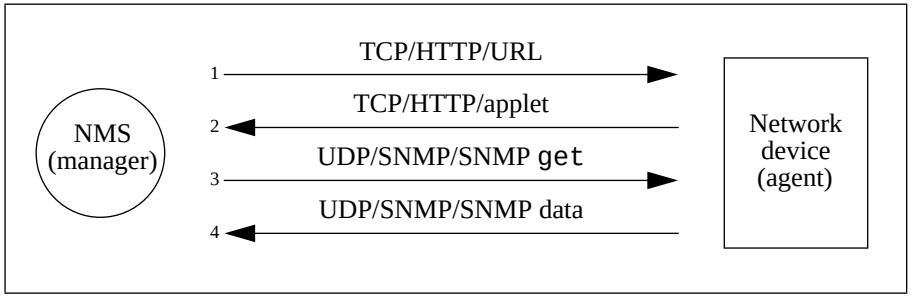
- Gestion *ad hoc* :
  - interfaces graphiques propriétaires codées sous forme d'applettes
    - HTTP en plus de SNMP [Bruins 1996]
    - HTTP au lieu de SNMP [Wellens and Auerbach 1996]
  - interfaces graphiques génériques codées sous forme d'applettes
- Gestion régulière :
  - toutes les interfaces graphiques sont codées sous forme d'applettes
  - *polling* basé sur HTTP

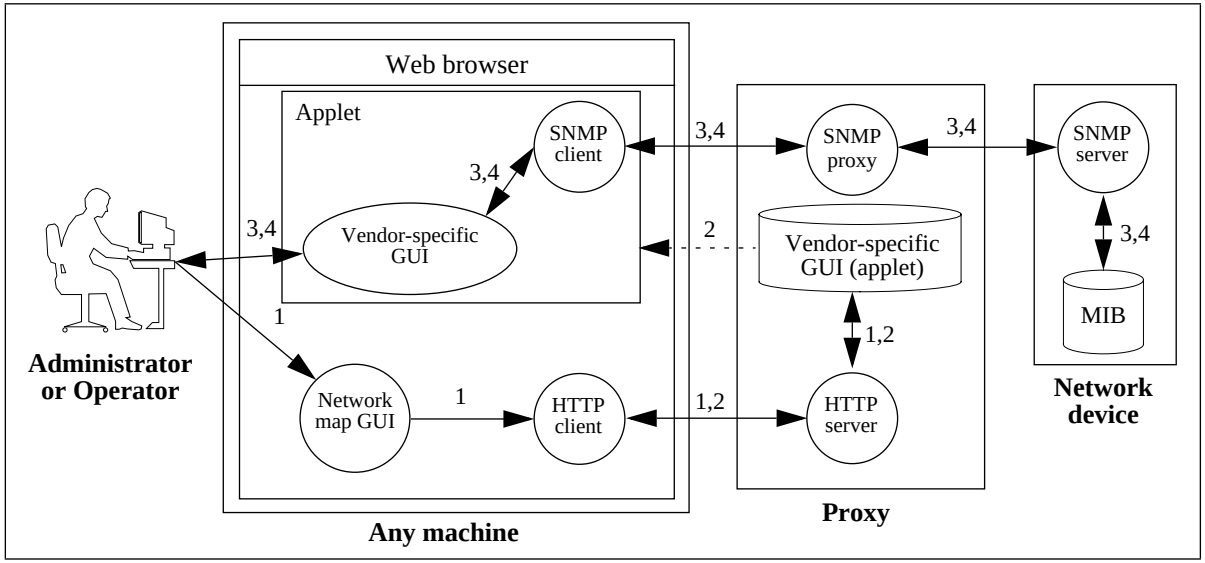
Pull *ad hoc* : HTTP en plus de SNMP (1/2)



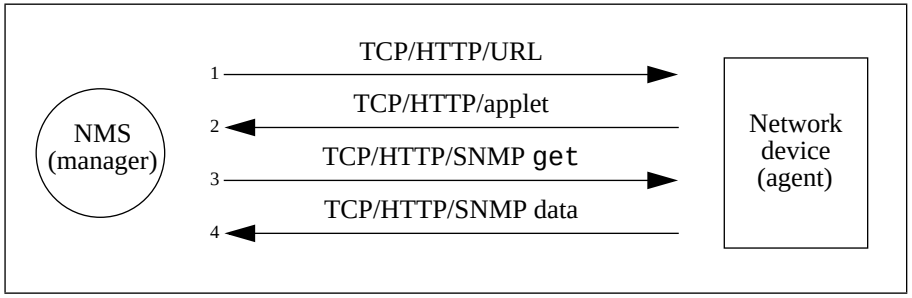


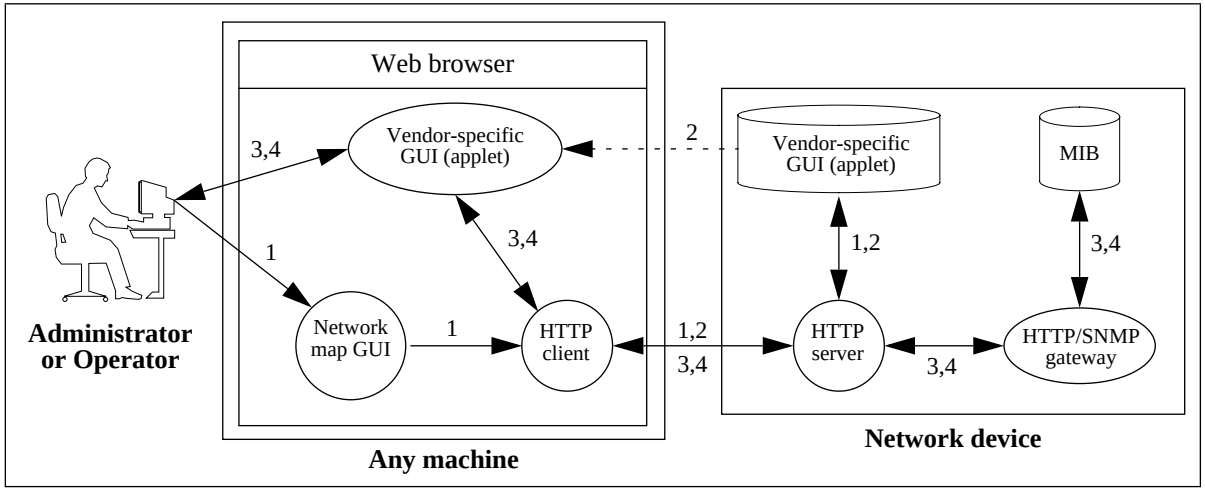
Pull *ad hoc* : HTTP en plus de SNMP (2/2)



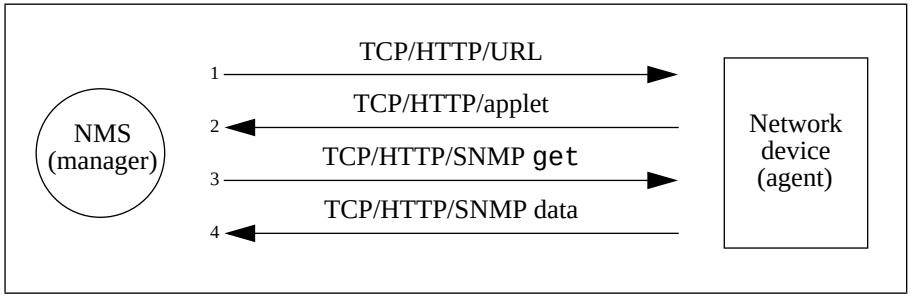


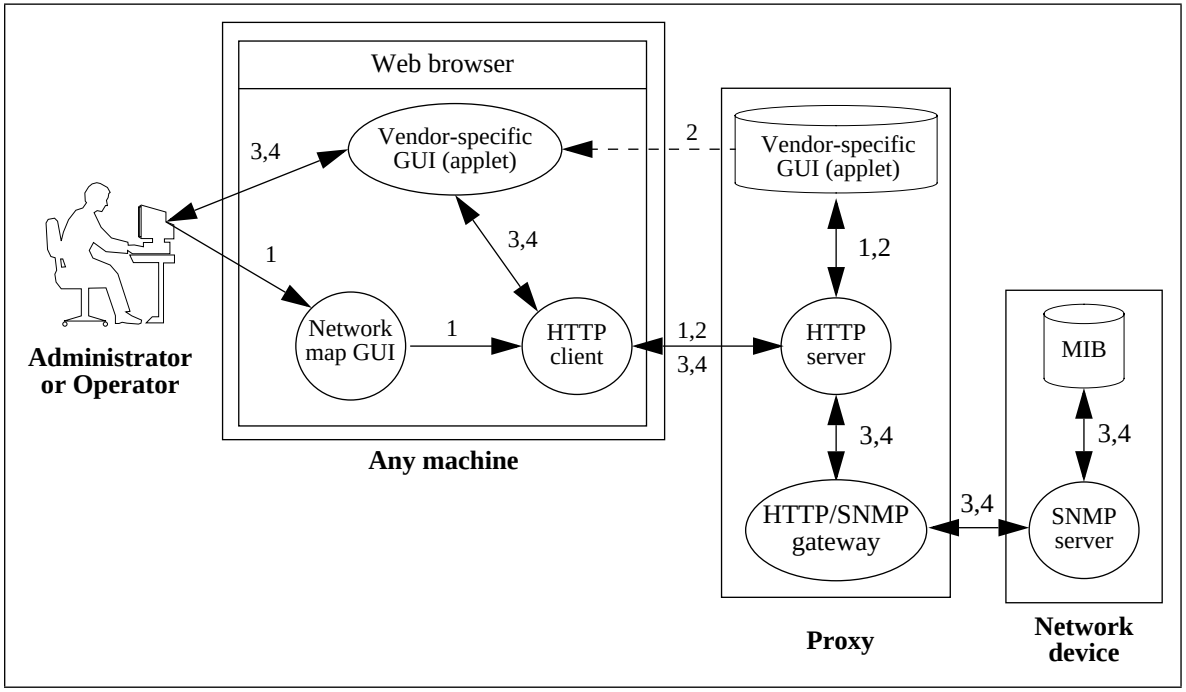
Pull *ad hoc* : HTTP au lieu de SNMP (1/2)



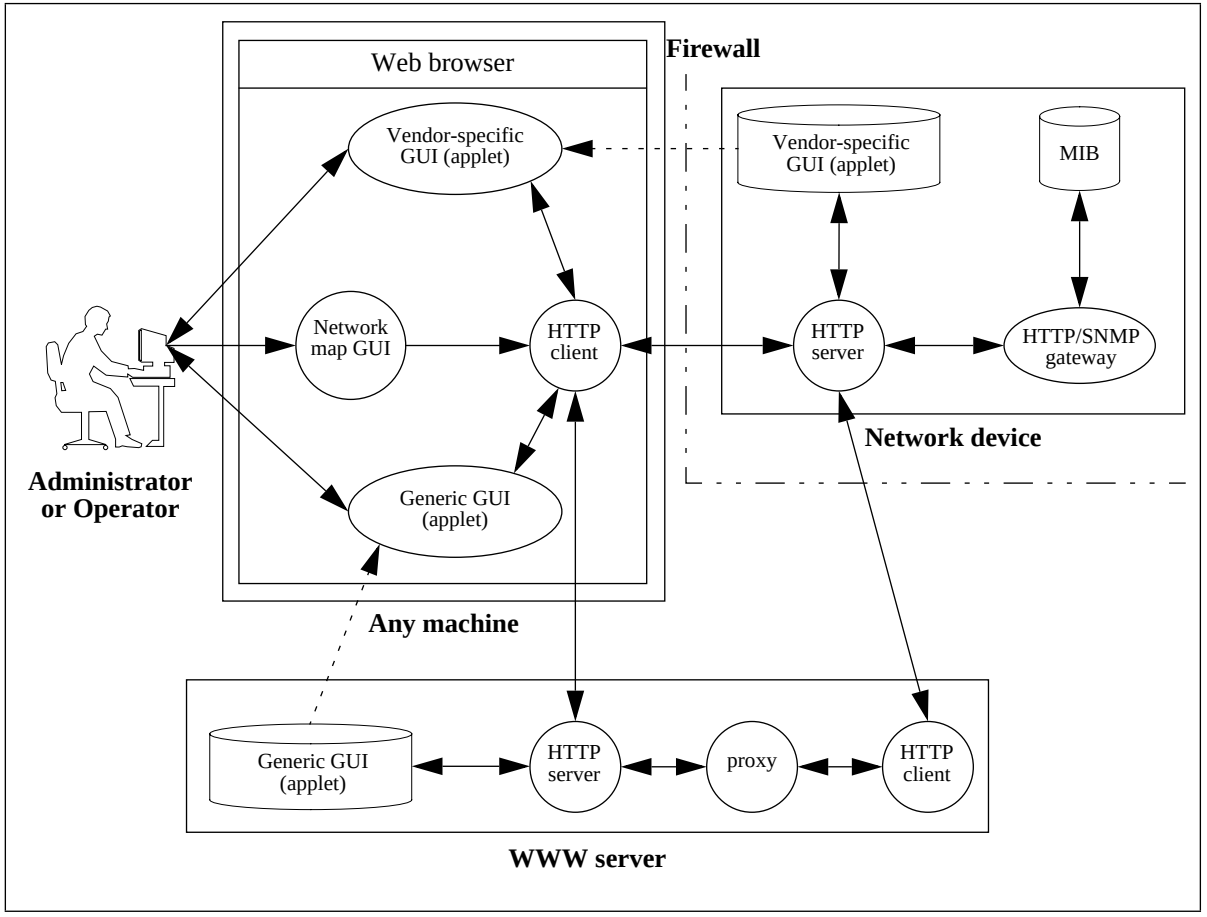


Pull *ad hoc* : HTTP au lieu de SNMP (2/2)





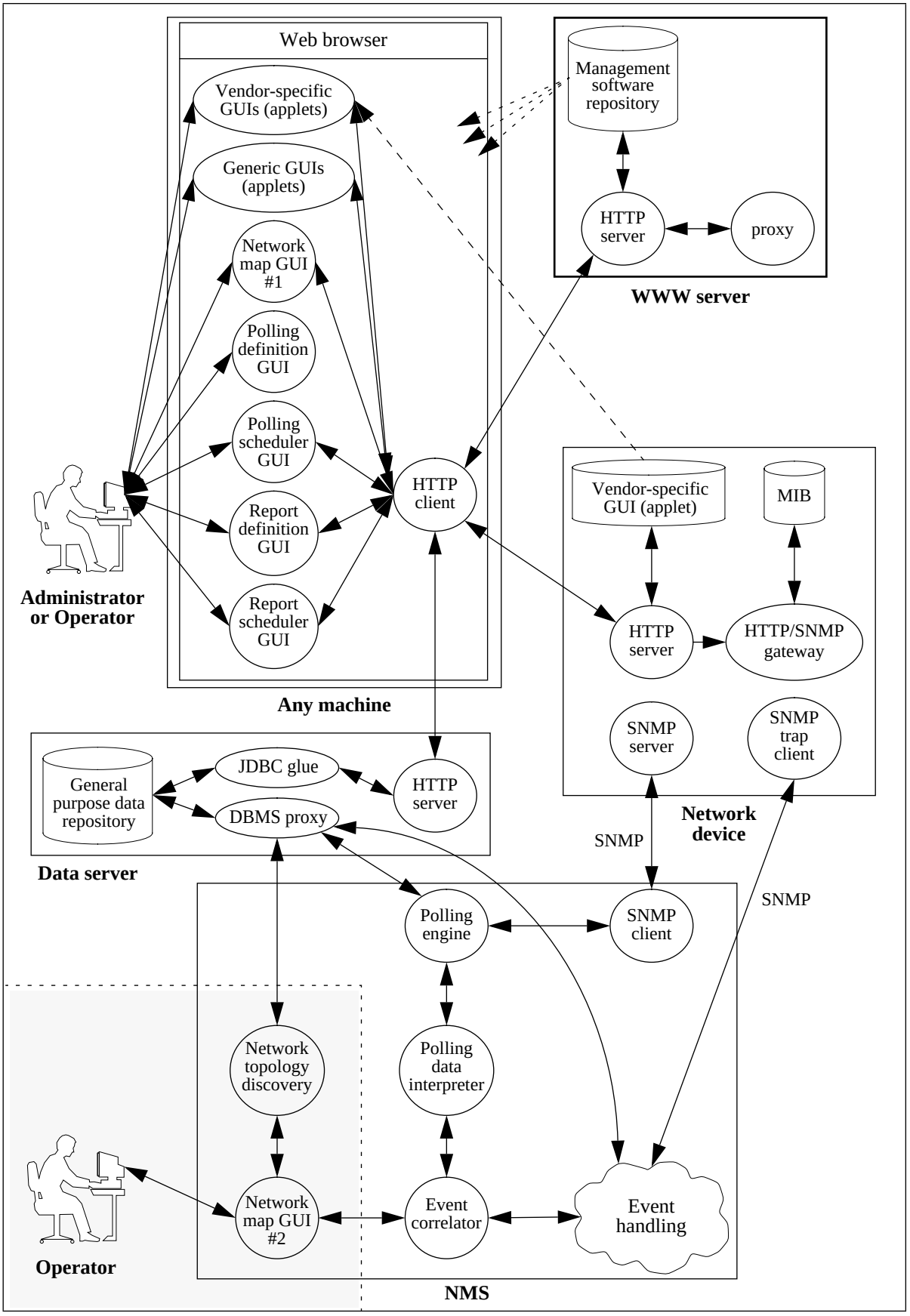
Pull *ad hoc* : interfaces graphiques génériques codées sous forme d'applettes



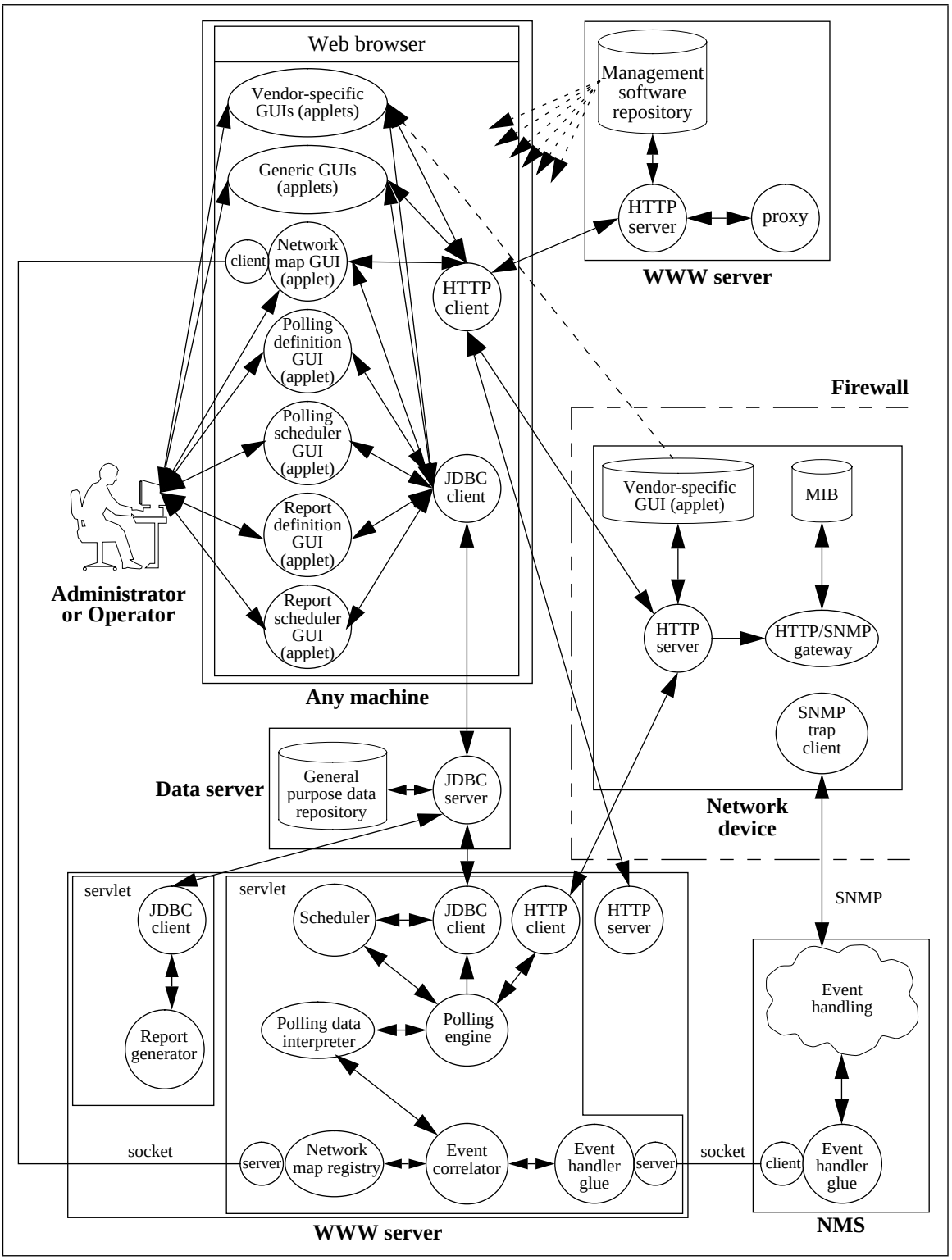
Modèle *pull*

- Gestion *ad hoc* :
  - interfaces graphiques propriétaires codées sous forme d'applettes
    - HTTP en plus de SNMP [Bruins 1996]
    - HTTP au lieu de SNMP [Wellens and Auerbach 1996]
  - interfaces graphiques génériques codées sous forme d'applettes
- Gestion régulière :
  - toutes les interfaces graphiques sont codées sous forme d'applettes
  - *polling* basé sur HTTP

Toutes les interfaces graphiques codées sous forme d'applettes



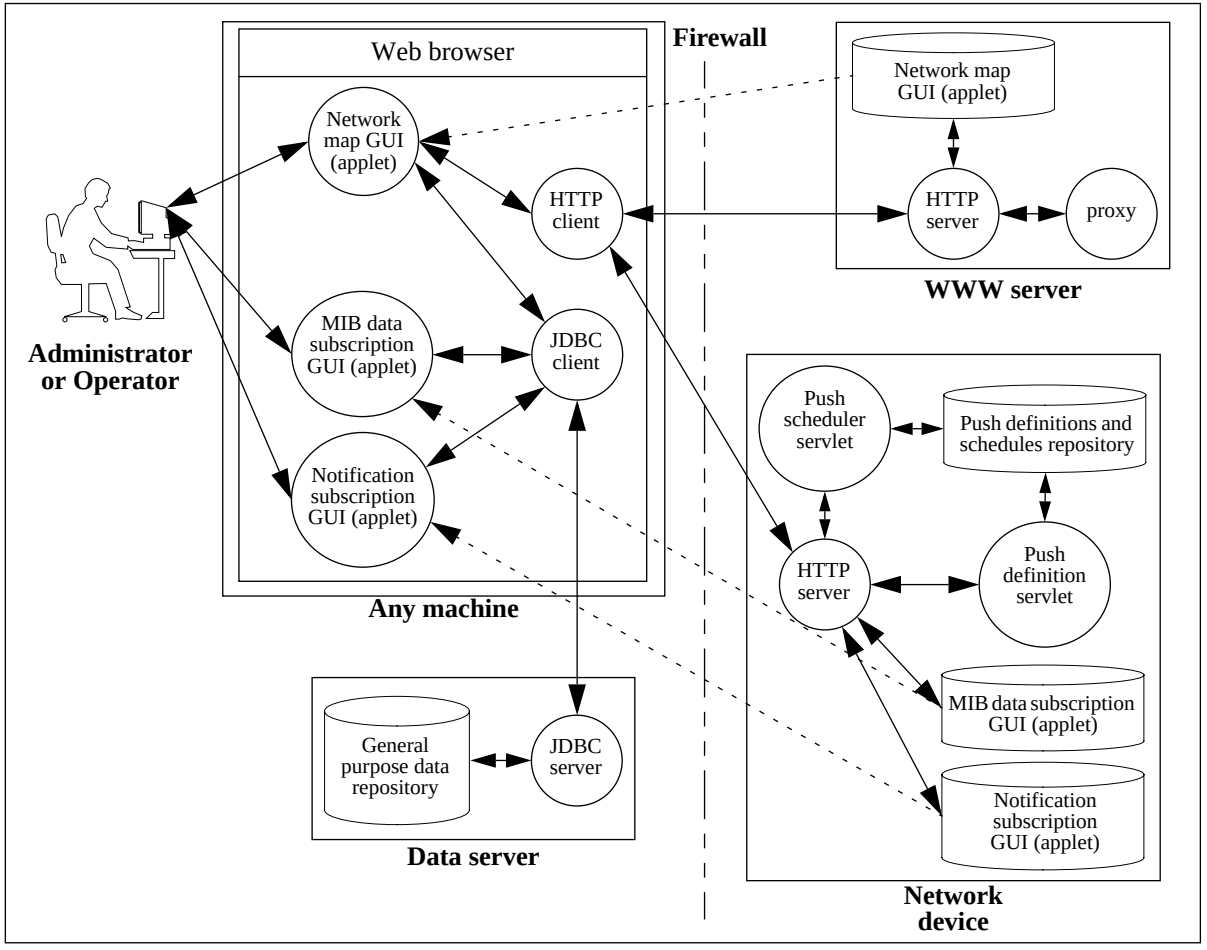
Polling basé sur HTTP



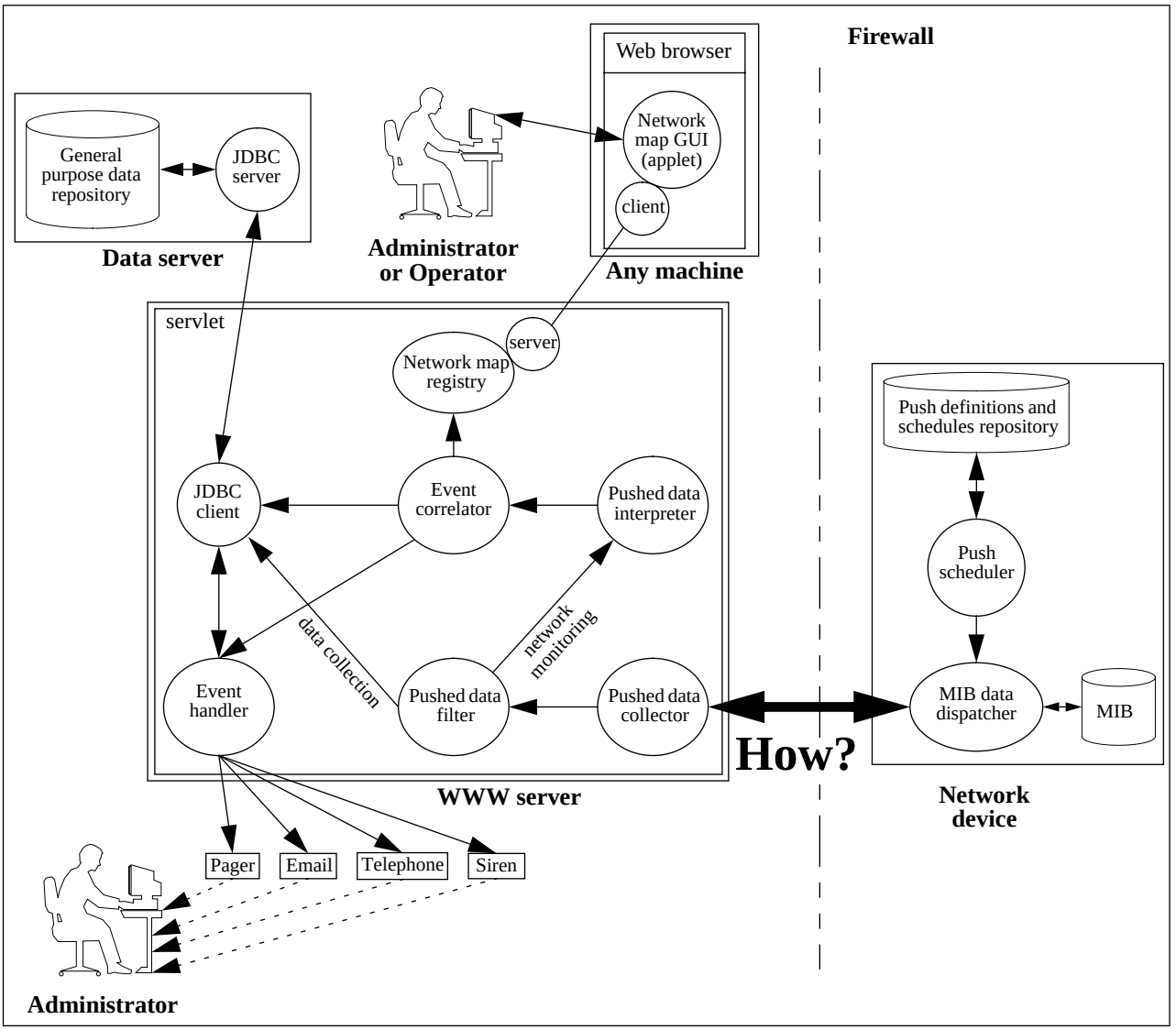
Le modèle *push*

- Gestion régulière
- 3 phases :
  - publication
  - souscription
  - distribution

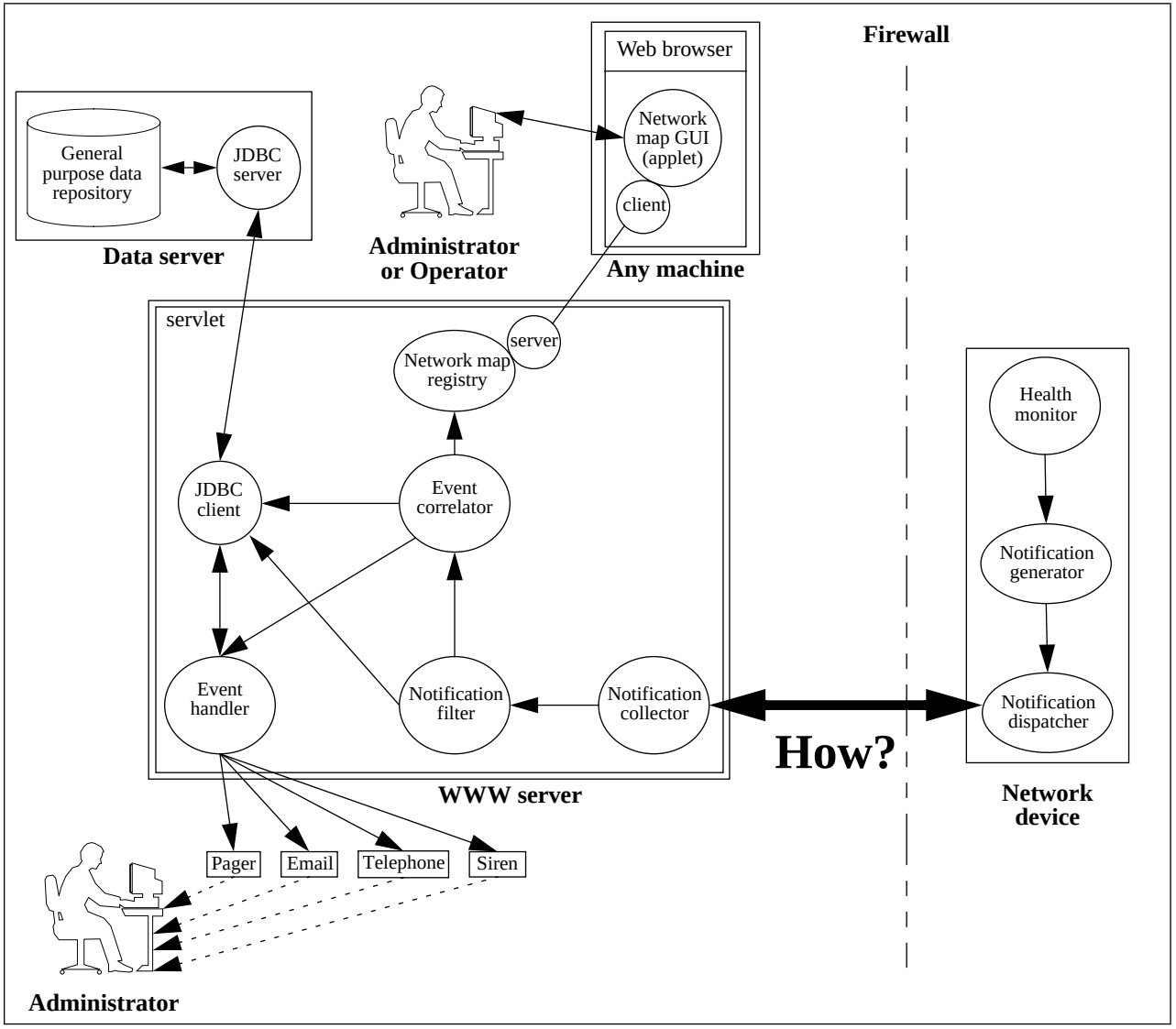
Phases "publication" et "souscription"



Phase “distribution” pour le monitoring et la collecte de données



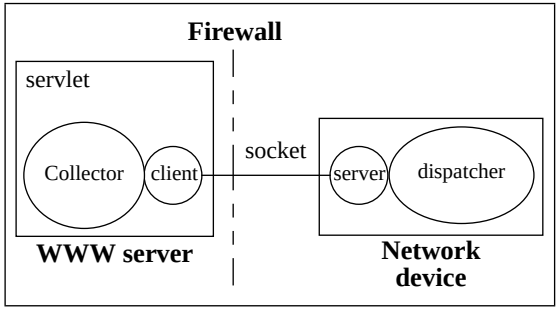
Phase "distribution" pour les notifications



Problèmes

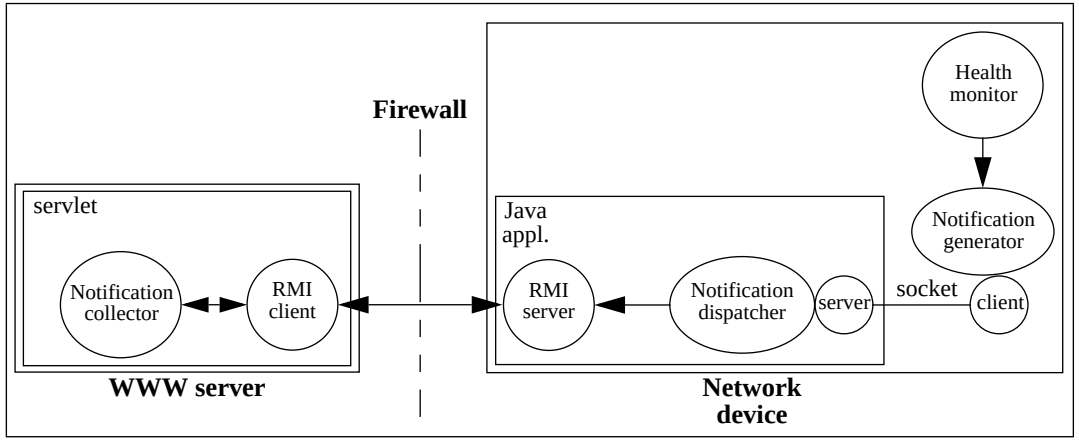
- Positions du client et du serveur inversées :
  - transfert des données de gestion initié par l'agent
  - côté client de la connexion persistante toujours du côté du gestionnaire (sécurité)
  - on a besoin d'initier une communication de type client-serveur depuis le serveur : problème !
- *Firewalls* : HTTP, *sockets* ou RMI ?
- *Timeout* d'une connexion persistante

Socket

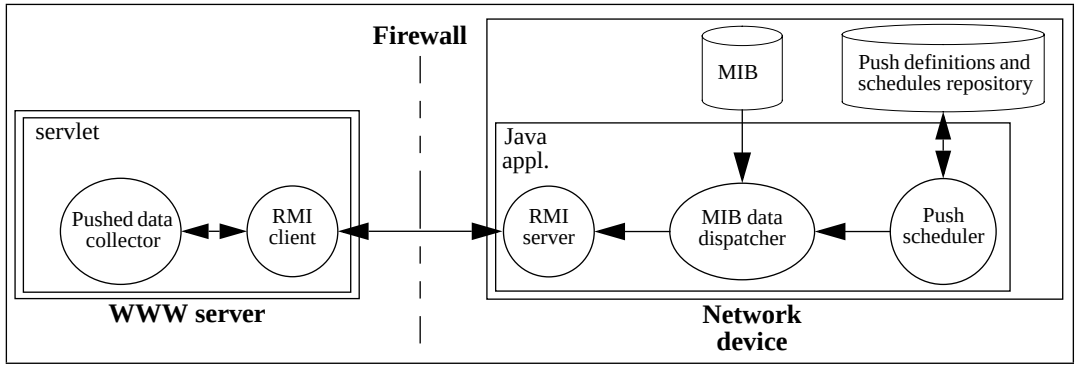


- Avantages :
  - association bidirectionnelle
  - simple à implémenter
- Inconvénients :
  - instable si *timeout* de la connexion TCP persistante < période de *push*
  - robustesse : l'envoi des notifications par l'agent dépend de la disponibilité d'une connexion TCP persistante créée par une autre entité (le gestionnaire)
  - le *firewall* nécessite une configuration spéciale (UDP ou TCP)

Java RMI (1/2)



Distribution via Java RMI pour les notifications

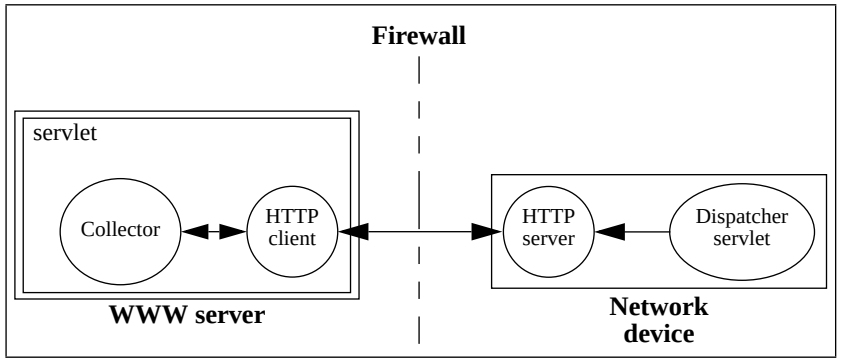


Distribution via Java RMI pour le monitoring et la collecte de données

Java RMI (2/2)

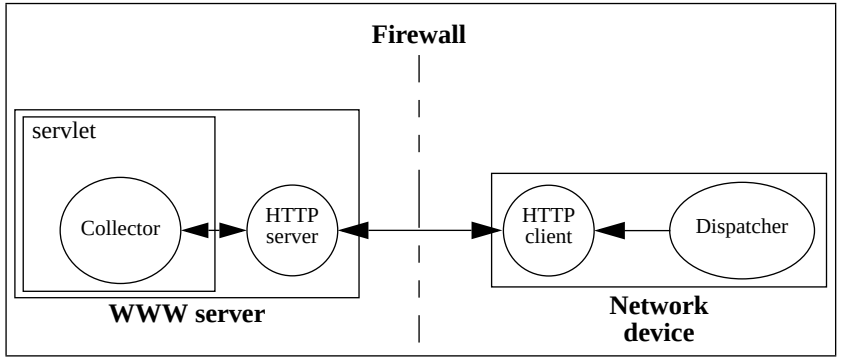
- Avantages :
  - association bidirectionnelle (car RMI utilise des *sockets*)
  - conception élégante (gestion entièrement orientée objets)
- Inconvénients :
  - nécessite une JVM complète dans les agents (irréaliste pour de nombreux équipements)
  - l'exécution de code Java est lente, ce qui pose un problème d'échelle (*scalability*)
  - *firewalls* : comment contrôler les ports utilisés par les clients RMI (ils sont supposés être transparents pour l'application) ?

HTTP (1/1)



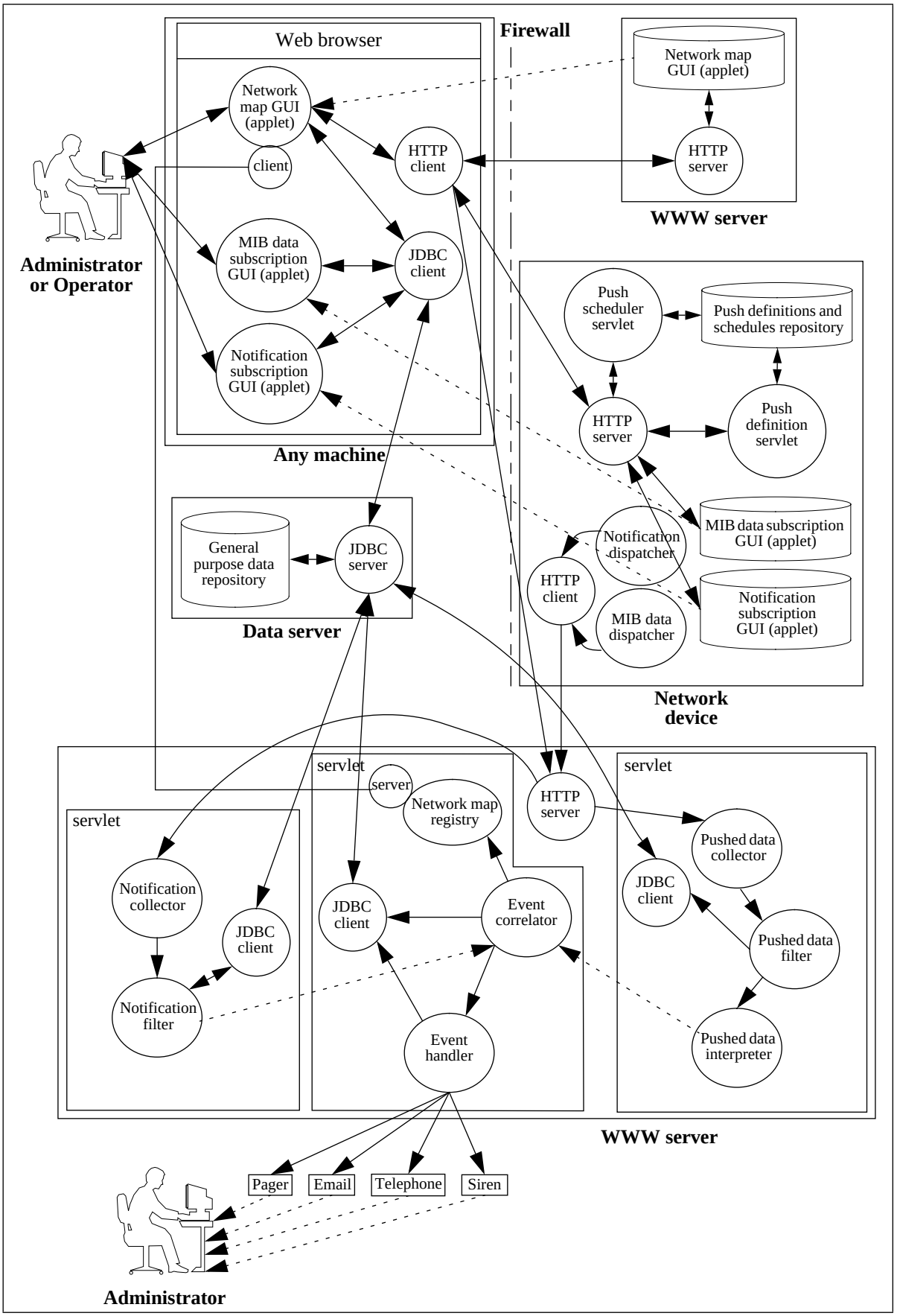
- Message MIME (Multipurpose Internet Mail Extensions) comportant un nombre infini de parties [Netscape 1995]
- Avantages :
  - simple à implémenter
  - *firewalls* : aucun ou peu de changements (si déjà accès Web)
- Inconvénients :
  - il faut pouvoir contrôler le *timeout* du serveur HTTP (Apache = OK)
  - il faut envoyer des *keepalives* pour détecter des pannes réseau

HTTP (2/2)



- Avantages :
  - modèle conceptuel simple : le client comme le serveur sont du bon côté
  - robustesse : l'agent peut se reconnecter aussitôt en cas de *timeout*, sans avoir à compter sur le gestionnaire
  - *firewalls* : aucun ou peu de changements (si déjà accès Web)
- Inconvénients :
  - connexion créée par une entité externe (usurpation d'identité...)

Push HTTP : vue d'ensemble



Problèmes résolus (1/5)

- Clients :
  - les PGR coûtent trop cher (matériel et logiciel) :
    - ➠ plus besoin de matériel dédié à la gestion de réseaux
    - ➠ logiciel moins cher (applettes et servelettes Java, plus quelques scripts)
    - ➠ meilleure utilisation des investissements précédents (pas besoin de racheter une BD relationnelle ou de payer le portage pour un PGR donné)
  - le support de BD relationnelles tierce-parties est trop limité
    - ➠ avec JDBC, plus besoin de compter sur des accords bilatéraux entre vendeurs de PGR et de BD relationnelles
  - besoin d'expertise Unix pour la maintenance des PGR existants :
    - ➠ migration aisée d'une plateforme (PC, Unix, Mac) à une autre : Java et butineur Web indépendants de la plateforme

Problèmes résolus (2/5)

- Vendeurs de matériel :
  - coûts de développement des *add-ons* trop élevés :
    - une seule applette
- Clients et vendeurs de matériel :
  - temps de mise sur marché des *add-ons* trop long :
    - réduit à zéro
    - compétition loyale entre tous les vendeurs, quelle que soit leur part de marché
  - gestion de versions multiples d'une MIB :
    - les MIB et les applettes sont mises à jour simultanément, agent après agent
    - on peut utiliser les nouvelles fonctionnalités d'une MIB aussitôt

Problèmes résolus (3/5)

- Efficacité du protocole :
  - inefficacité de l'encodage BER :
    - ➠ plus utilisé
  - inefficacité de SNMP :
    - ➠ le protocole de communication SNMP est remplacé par HTTP
  - réduction de l'impact des données sur la BP du réseau (*network overhead*) :
    - ➠ les données de gestion peuvent être compressées de façon transparente (SNMPv3 le permet aussi, mais pas SNMPv1 ni SNMPv2c)
    - ➠ *overhead* des entêtes de messages plus faible :
      - nombreuses données par message (une seule *varbind list* avec SNMPv1, SNMPv2c et SNMPv3)

Problèmes résolus (4/5)

- Sécurité :
  - gestion d'une filiale à distance (VPN) : sécurité de HTTP
    - faible niveau de sécurité
    - mieux que *community string* de SNMPv1 et SNMPv2c
  - *firewalls* : HTTP plus simple que SNMP (plus de relais UDP)
- Protocole de transport :
  - HTTP utilise TCP
  - protocole de transport fiable
  - toujours pas de garantie d'arrivée à destination

Problèmes résolus (5/5)

- Sémantique :
  - XML :
    - bien plus d'opérations possibles qu'avec SNMP
    - des structures de données plus complexes sont possibles
  - Java RMI :
    - niveau d'abstraction plus élevé
    - entièrement orienté objets --> API de haut niveau

Bonus

- Des gestionnaires redondants sont simples à supporter avec le modèle *push* :
  - multicast IP
    - ➠ redondance transparente pour l'agent
  - plusieurs souscripteurs
    - ➠ redondance explicite
  - une étape vers la tolérance de pannes

Nouveaux problèmes

- Vendeurs de PGR :
  - perte de revenus (notamment marché des PME)
  - nouvelles niches de marché :
    - ➠ tolérance de pannes
    - ➠ grands réseaux où les effets d'échelle sont critiques
    - ➠ réseaux temps réel où le temps de réaction et la BP sont primordiaux
- Clients et vendeurs de matériel :
  - problèmes connus :
    - ➠ synchronisation des horloges
    - ➠ besoin de contrôler le *timeout* d'une connexion persistante
  - problème à investiguer :
    - ➠ la vitesse d'exécution du code Java est lente (même avec JIT) :
      - quel impact sur la latence et la *scalability* ?

Recherche actuelle

- Implémenter les modèles *pull* et *push* décrits auparavant
- Démontrer la faisabilité et le potentiel de la gestion de réseaux basée sur les technologies Web
- On utilise pour ça :
  - AdventNet's SNMP package
  - Apache et Jigsaw
  - Sun's JMAPI
- On étudie :
  - Sun's Java DMK (M-beans)
  - Sun's EmbeddedJava

Recherche future

- Comparer les performances relatives des modèles *pull* et *push*
- Étudier les problèmes d'échelle et de performance
- Prototype : routeurs IP de Lightning

Pour en savoir plus

J.P. Martin-Flatin. *The Push Model in Web-Based Network Management*. Rapport technique SSC/1998/023, version 3, SSC, EPFL, Lausanne, Suisse, novembre 1998. Soumis à *ACM Computer Communication Review*, décembre 1998.

J.P. Martin-Flatin. *Push vs. Pull in Web-Based Network Management*. Rapport technique SSC/1998/022, version 3, SSC, EPFL, Lausanne, Suisse, novembre 1998. À paraître dans *Proc. 6th IFIP/IEEE International Symposium on Integrated Network Management (IM'99)*, Boston, MA, USA, mai 1999.

J.P. Martin-Flatin. *IP Network Management Platforms Before the Web*. Rapport technique SSC/1998/021, version 2, SSC, EPFL, Lausanne, Suisse, décembre 1998.

Ces transparents sont disponibles depuis:

http://ica2www.epfl.ch/~jpmf/talks/imag_990211.pdf