

Une Architecture Multi-Observateur pour l'Observation des Services Web Composés

Abdelghani Benharref, Mohammed Adel Serhani,
Roch Glitho, Rachida Dssouli
Concordia University



Outline

- Introduction \ Contexte
- Architecture pour SW simples
- Besoins en matière d'observation
- Modèles de fautes
- Architecture
- Implémentation
- Conclusion



Idées (questions) de base

- ❑ Comment **tester** un service web dans son **environnement final** et **sans déranger** son fonctionnement?
- ❑ Comment le faire **en ligne** et de **manière automatique**?
- ❑ Comment **fournir ce service à tous les intervenants**?
- ❑ Comment **minimiser les ressources utilisées** pour des fins de test?

Implications

- **Transparence**
 - Pas de messages échangés avec le SW pour des fins de test
 - Minimiser les ressources/connaissances requis
 - Minimiser la charge réseau introduite par le test
- **Détection en ligne**
- **Le testeur doit être disponible pour le fournisseur du SW, son client ou même une entité mandatée**



Testeur passif

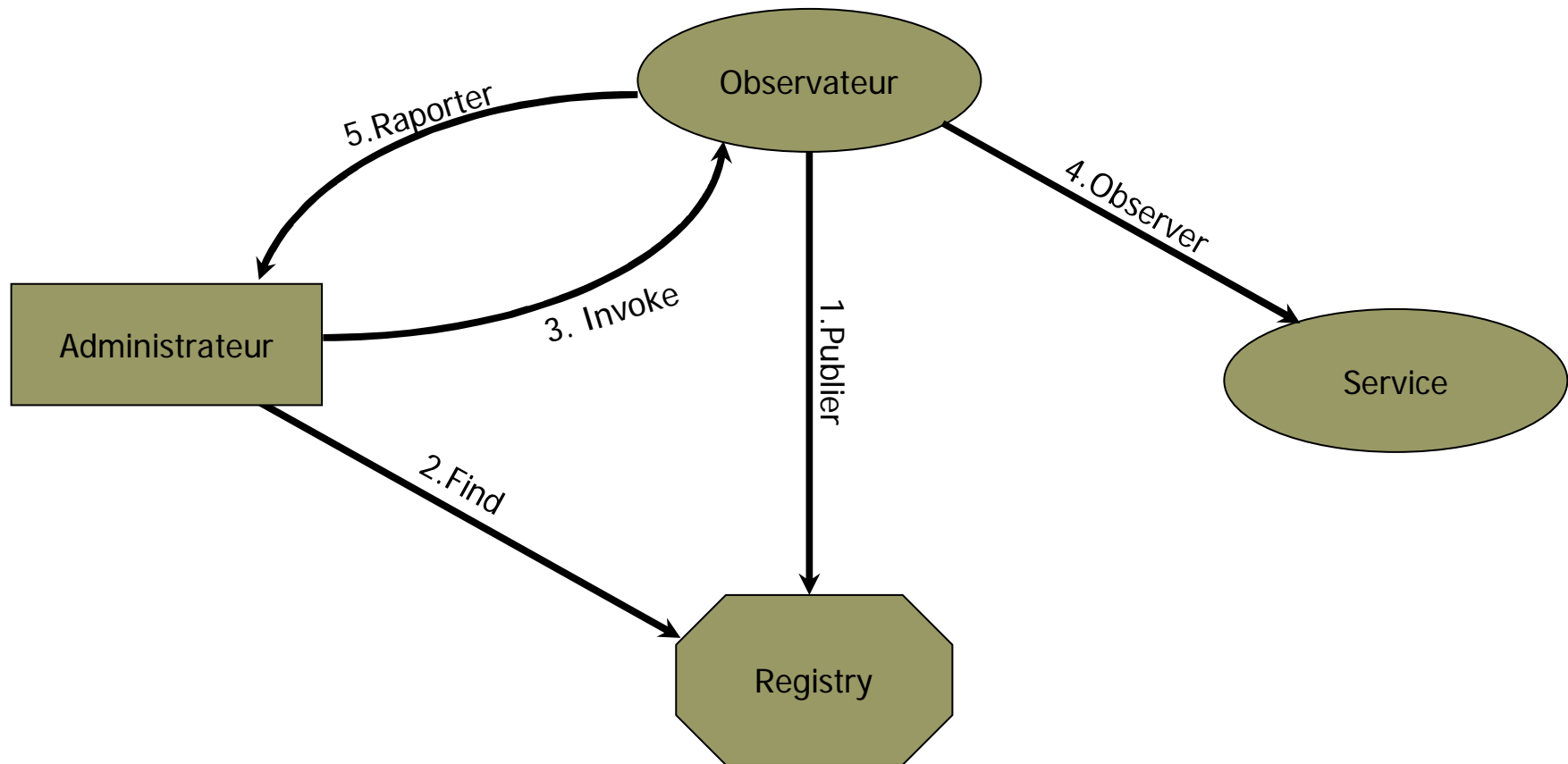


Profiter de
l'architecture des SW

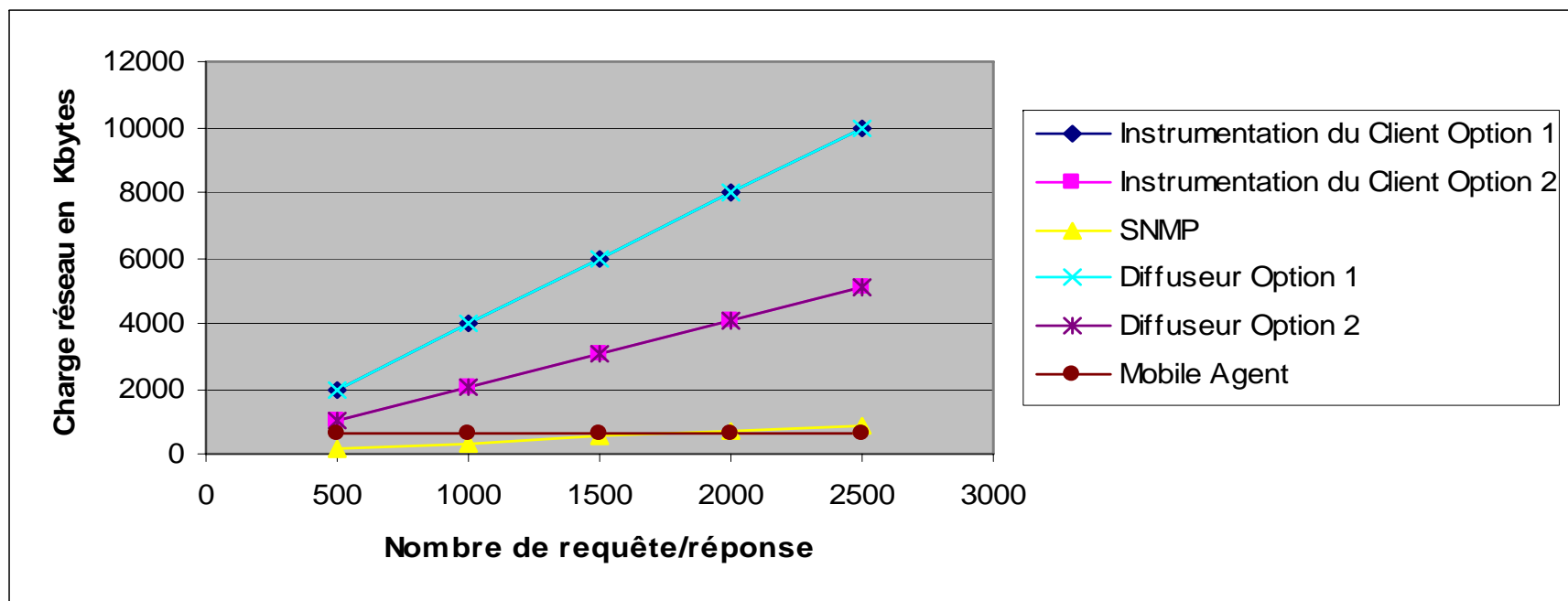


Un testeur passif
implémenté comme
SW

SW-Testeur

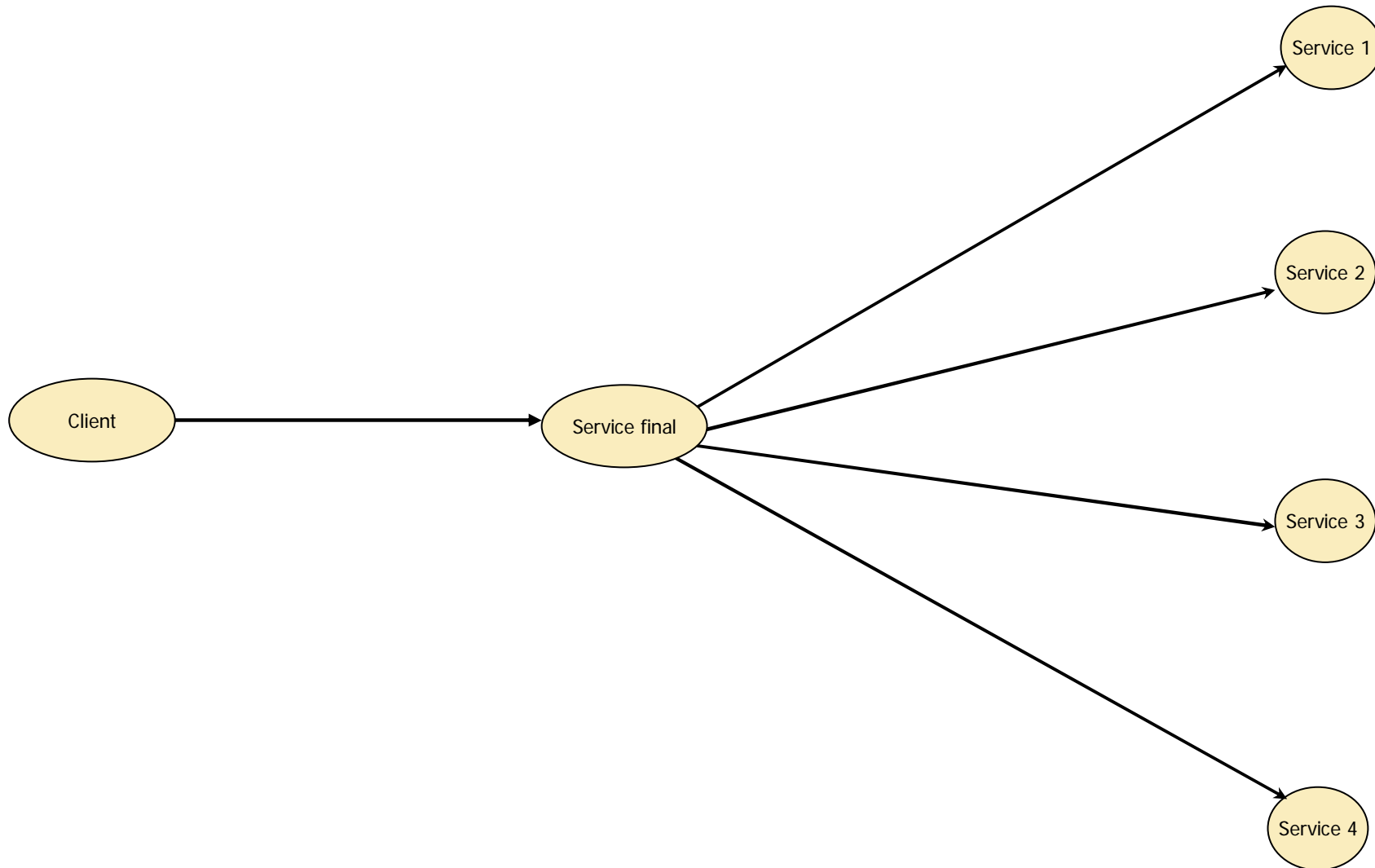


Collection de traces



Les agents mobiles présentent la charge réseaux minimale pour un nombre élevé de requête/réponse (à partir de 2000 req./res.)

Service web composé





Motivations

- Avoir une vue locale/globale sur les services en jeux
 - En cas de faute, trouver le service fautif
 - En cas d'un temps de réponse plus élevé, lequel des services est à améliorer
- Pouvoir utiliser l'information d'observation dans la sélection de service



Besoins pour l'observation

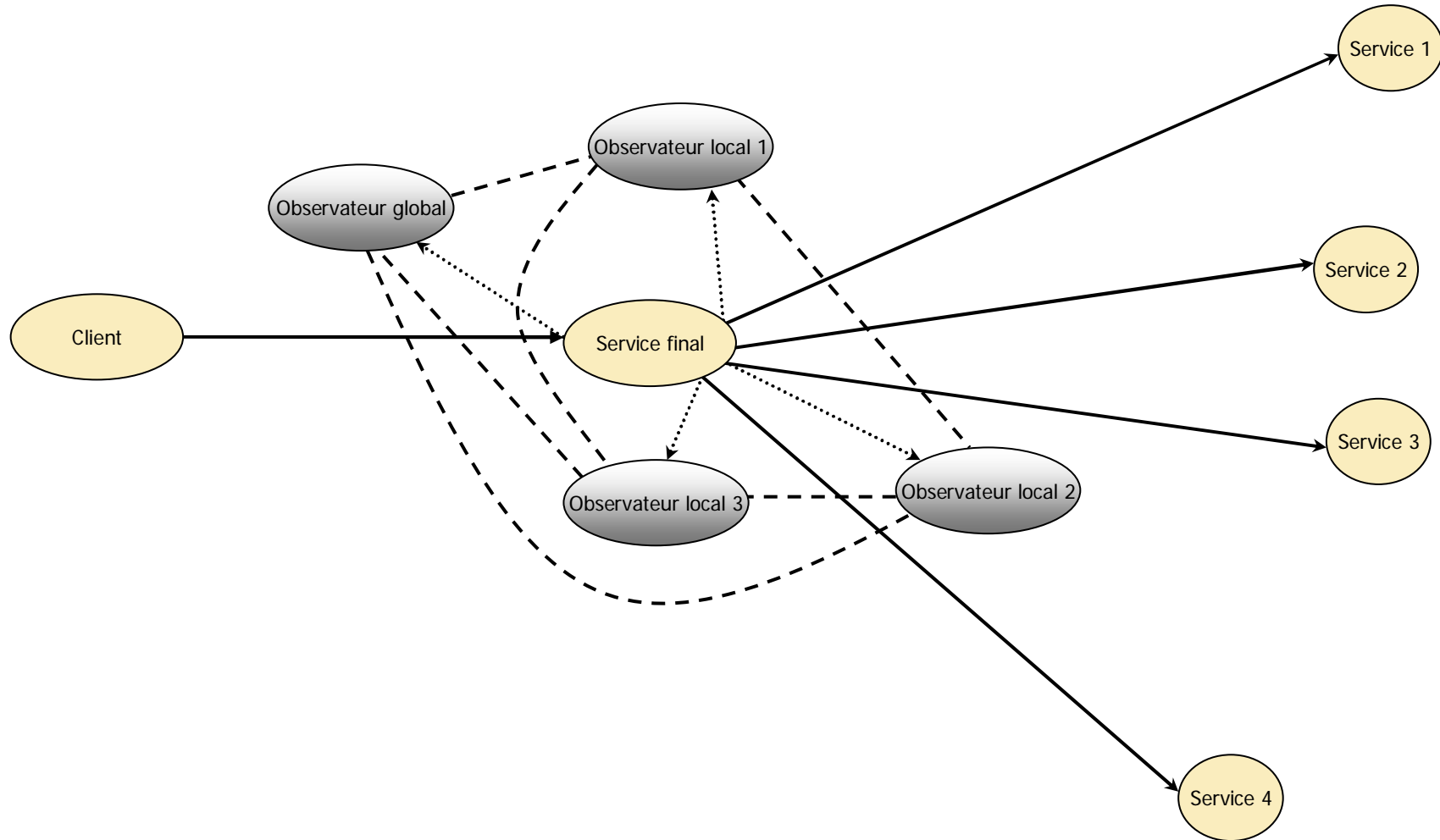
- ❑ BPEL du(es) service(s) composé(s)
- ❑ WSDL de tous les services
- ❑ Adresses (IP) exactes des services de base
- ❑ Modèles des services (FSM, FSM+temps,...)
- ❑ Plateforme pour agents mobiles



Modèles de fautes

- FSM/FSM+temps
 - Faute d'Entrée, Faute de Sortie
 - Dépassement de temps de réponse
- WSDL
 - Faute de Type d'Entrée
 - Faute de Type de Sortie
- BPEL
 - Faute d'Ordre (d'Orchestration)

Configuration 1: chez le service final

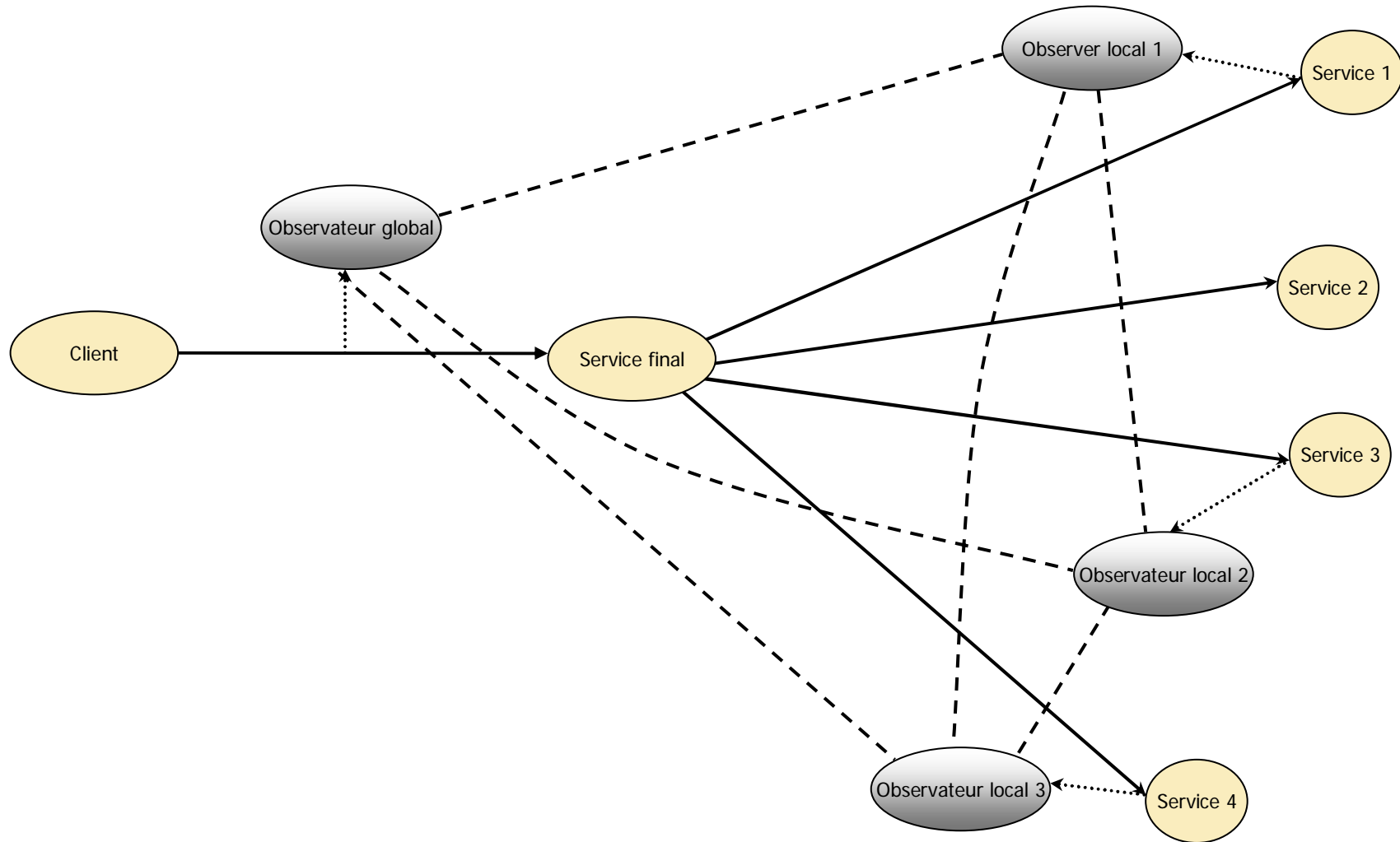




Avantages/Inconvénients

- Avantages:
 - Transparent aux services de bases
 - Un seul agent est généré/envoyé
 - Pas de trafic de coopération
 - Synchronisation facile
- Inconvénients
 - Le service final doit tout fournir

Configuration 2: chez tous les services

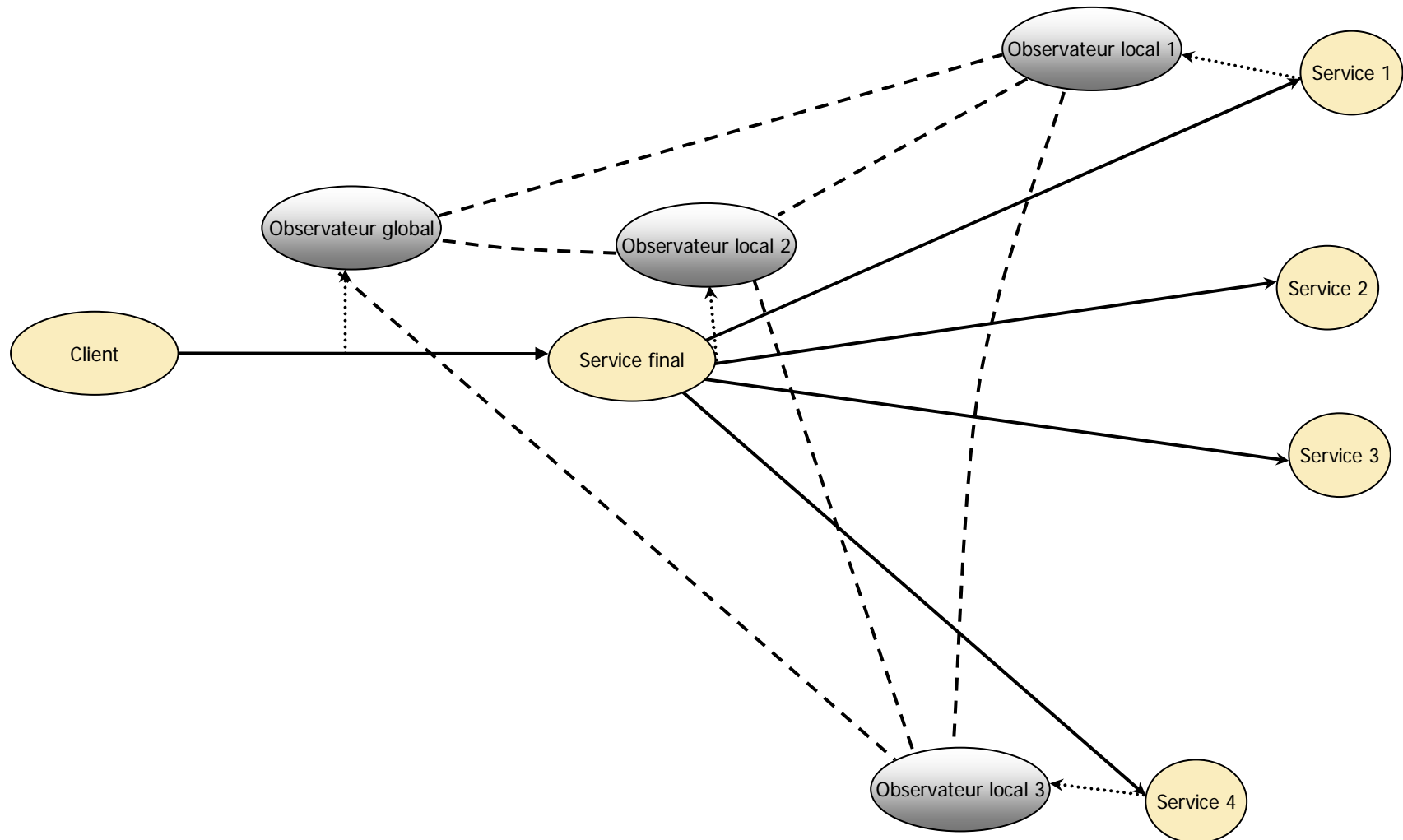




Avantages/Inconvénients

- Avantages
 - Chacun donne un coup de main
- Inconvénients
 - Tous les services doivent coopérer
 - Un agent distinct est envoyé à chaque service
 - Trafic de coopération des observateurs

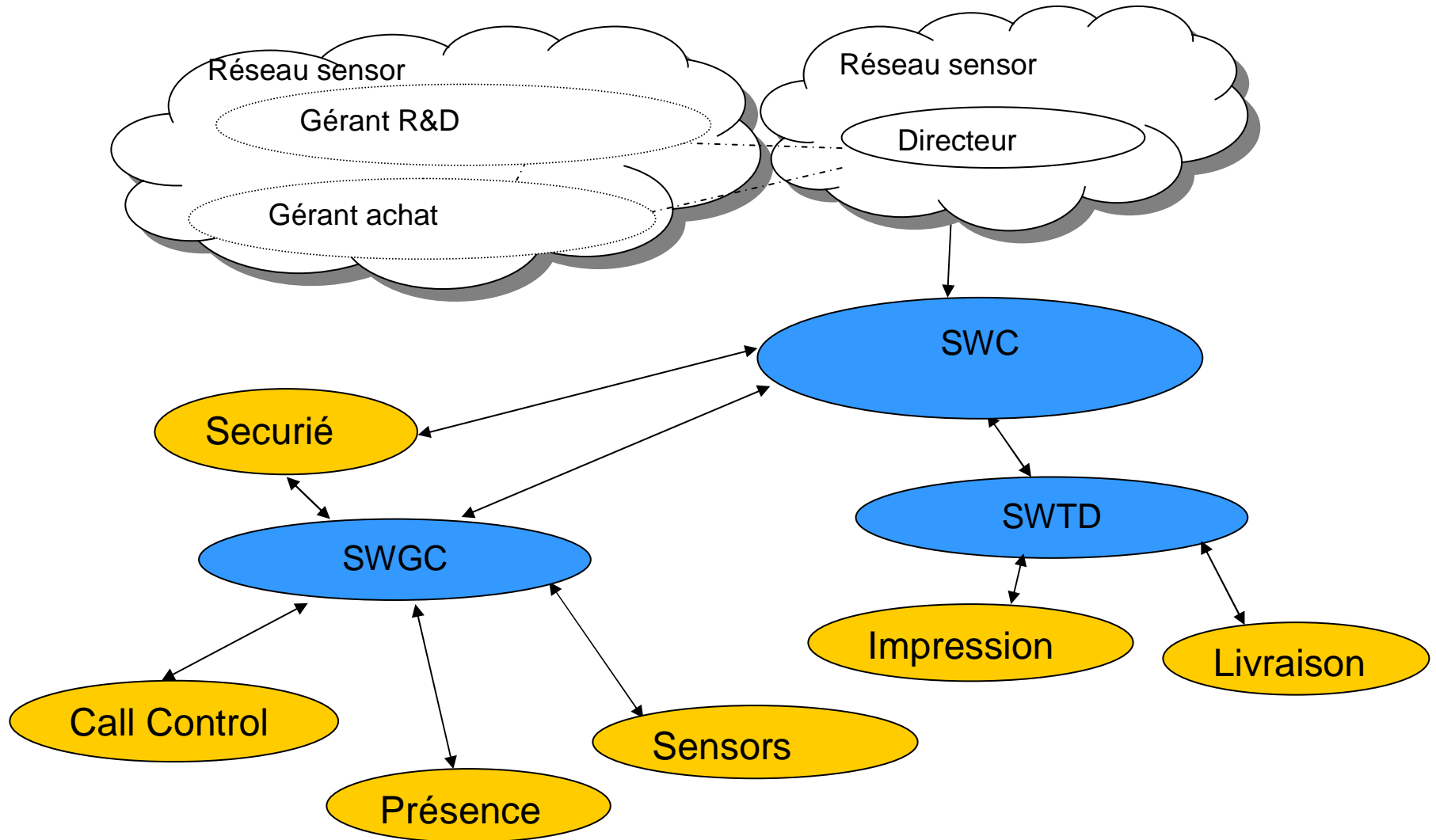
Configuration 3: hybride



Procédure (chez le service final)

- En 2 étapes:
 - Déploiement et configuration
 - Invocation du SW testeur
 - Le testeur génère un agent mobile et l'envoie
 - Clonage de l'agent
 - L'agent observant le service final devient l'observateur global
 - Observation
 - Reconnaissance d'états
 - Détection de faute

Implémentation





Conclusion/Travaux à venir

- Service simple → Observation simple
- Service composé → Observation composée
- Localiser un service fautif
- Terminer l'implémentation (très bientôt!)
- Étendre les observateurs avec un modèle EFSM/TEFSM