Programmation de services en téléphonie sur IP

Présentation de projet mémoire

Grégory Estienne

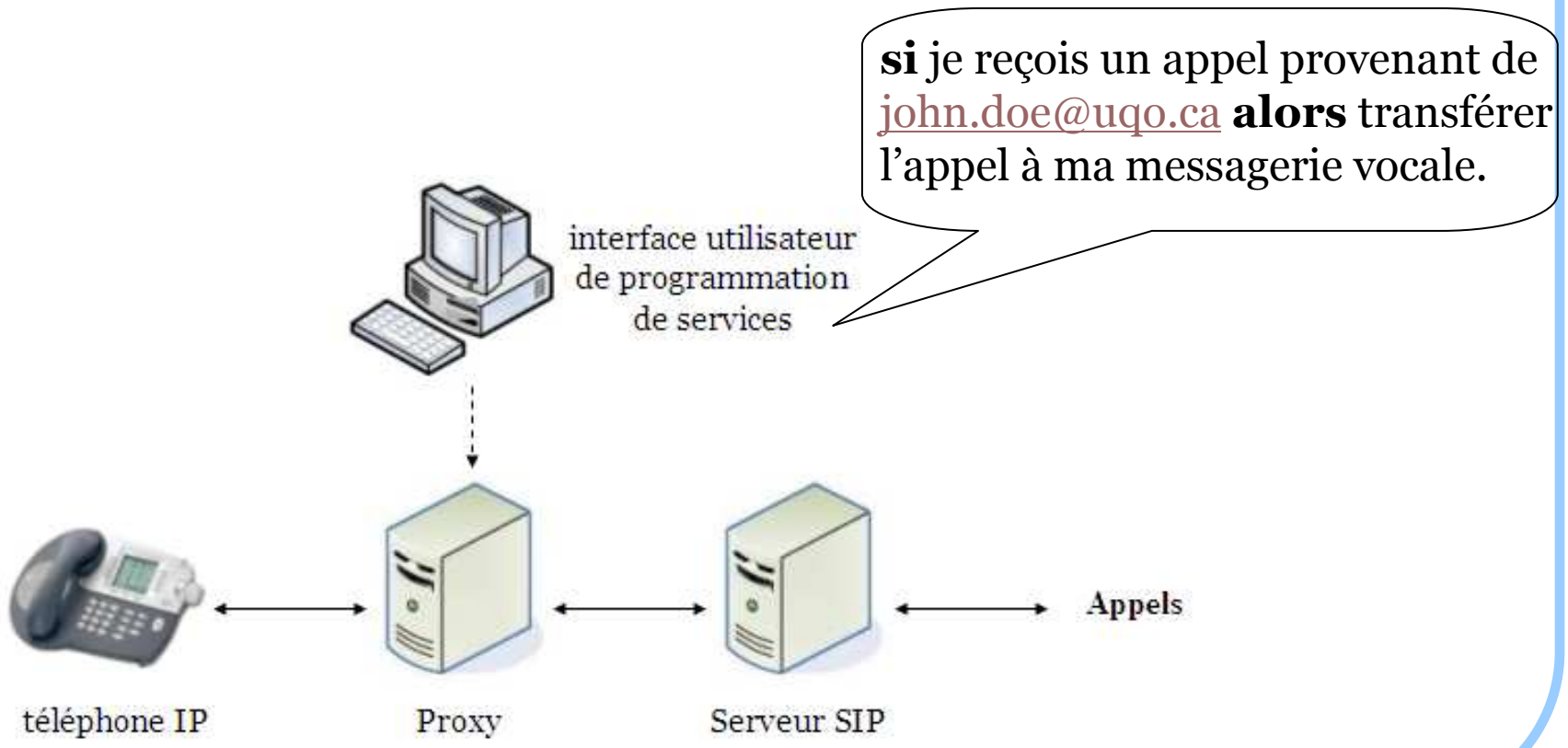
Sous la supervision du Dr. Luigi Logrippo

Introduction

- La téléphonie sur IP comme support à la programmation de services personnalisés
  - Architecture ouverte
  - Protocoles standardisés
  - Convergence des réseaux de communication
  - Cohabitation des différents médias : audio, vidéo, texte, ...
- Une nouvelle génération de services : les services sensibles au contexte (*context-aware services*)
  - Utilisation d'un contexte pour déterminer le comportement à adopter face à un événement

Introduction

- En quoi consiste la programmation de services ?



Objectifs du projet

Notre projet :

1. Proposer un modèle d'architecture adaptée aux services sensibles au contexte.

2. Concevoir et développer un système de programmation capable d'exprimer ces services (simulation).

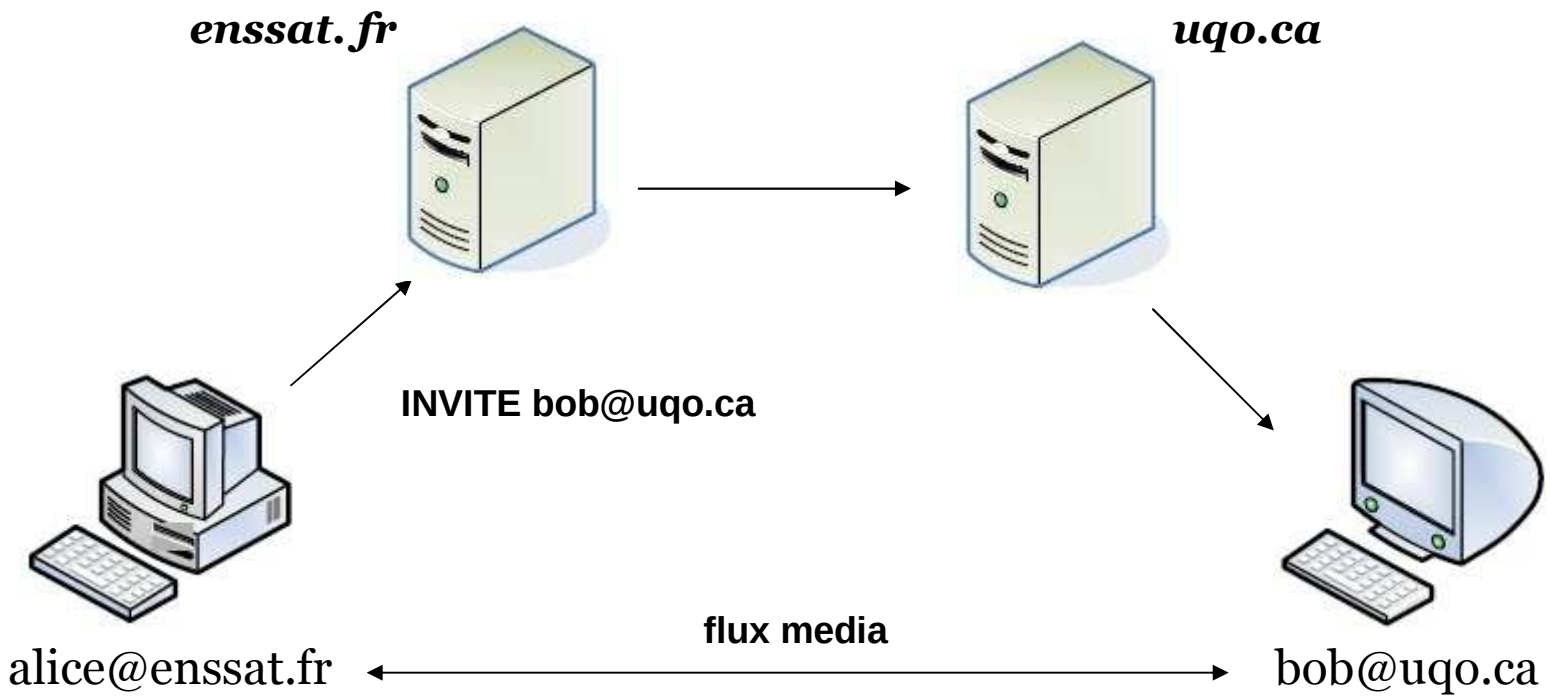
Plan de la présentation

1. SIP / SIMPLE
2. Revue de la littérature
3. Architecture de services
4. Programmation de services

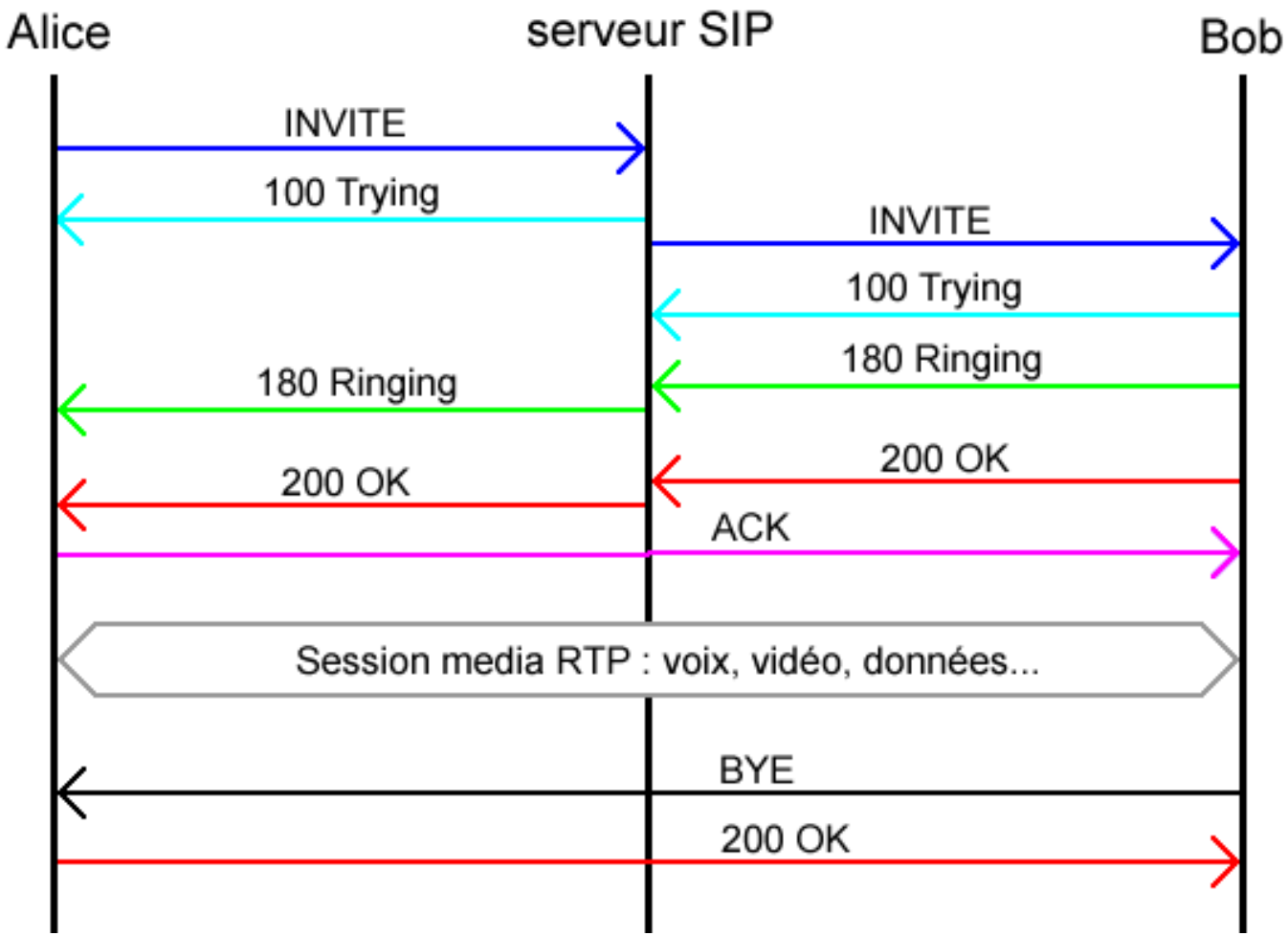
SIP

- **SIP** : Session Initiation Protocol, juin 2002 [RFC 3261]
  - protocole de signalisation qui permet d'établir, modifier et terminer des sessions multimédias.
  - Signalisation / flux média : 2 cheminements distincts.
- **SIMPLE** : SIP for Instant Messaging and Presence Leveraging Extensions
  - Nombreuses extensions à SIP, ex: le support de la messagerie instantanée et de la présence.
  - **Présence** : moyen d'exprimer la capacité et la volonté d'un individu à communiquer au travers d'un ensemble d'appareils.

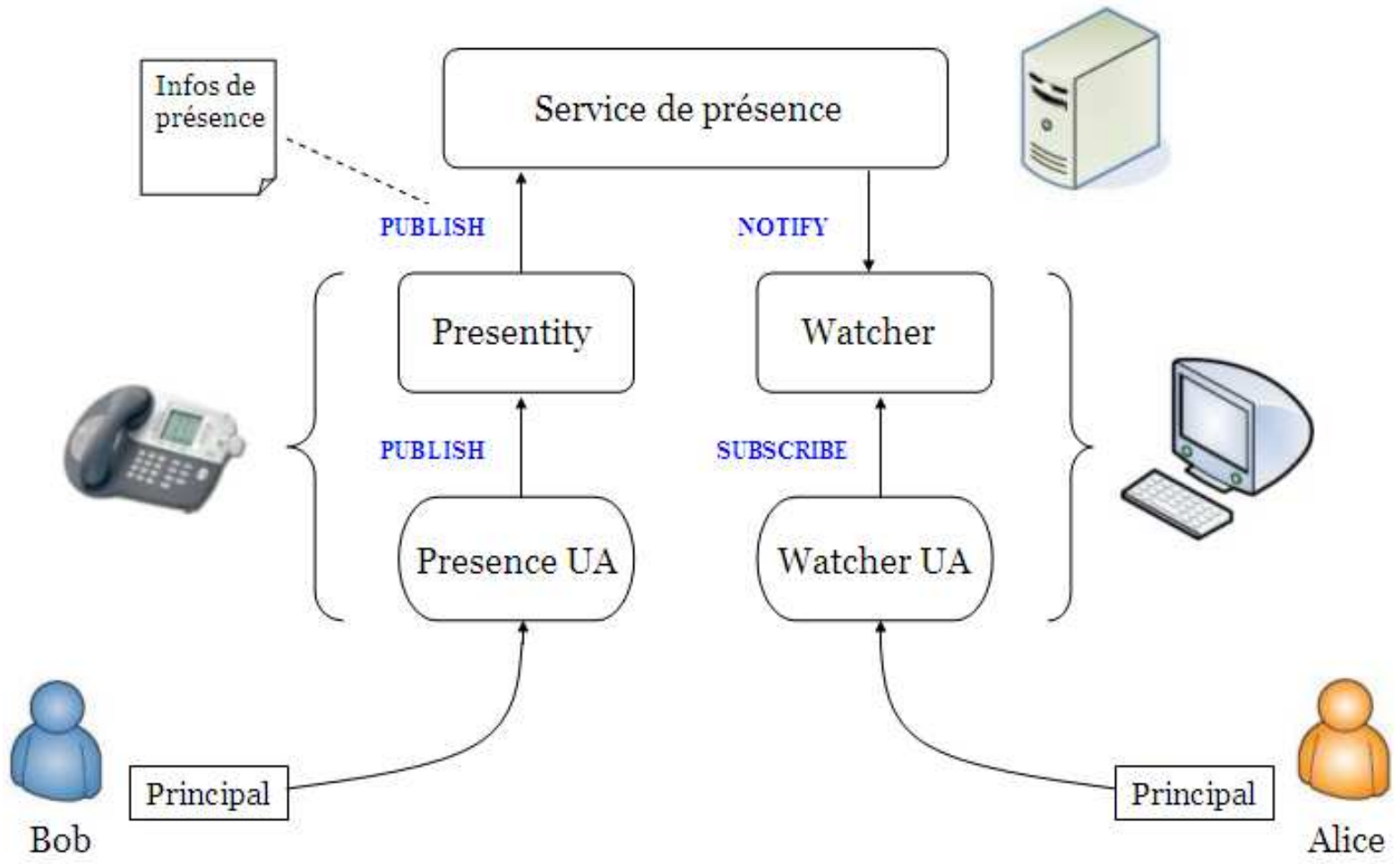
SIP


enssat.fr
uqo.ca
INVITE bob@uqo.ca
flux media
alice@enssat.fr
bob@uqo.ca

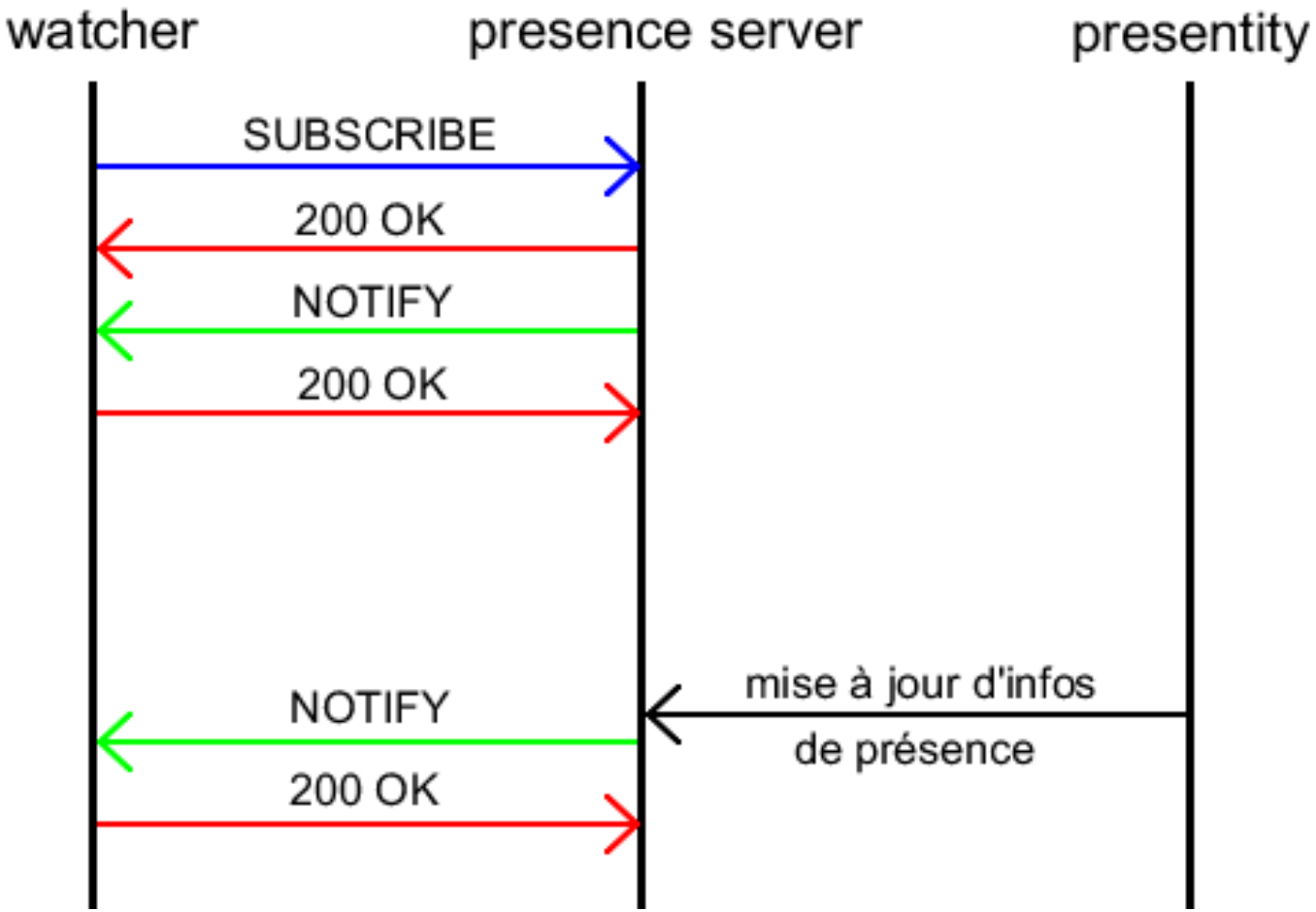
SIP


Alice
serveur SIP
Bob
INVITE
100 Trying
INVITE
100 Trying
180 Ringing
180 Ringing
200 OK
200 OK
ACK
Session media RTP : voix, vidéo, données...
BYE
200 OK

SIMPLE


Infos de présence
Service de présence
PUBLISH
NOTIFY
Presentity
Watcher
PUBLISH
SUBSCRIBE
Presence UA
Watcher UA
Principal
Principal
Bob
Alice

SIMPLE


watcher
presence server
presentity
SUBSCRIBE
200 OK
NOTIFY
200 OK
mise à jour d'infos
de présence
NOTIFY
200 OK

Revue de la littérature

- **IMS** : IP Multimedia Subsystem, 3rd Generation Partnership Project
  - Première présentation en mars 2003 (version 5 de l'UMTS).
  - Architecture visant à fournir des services multimédias quelle que soit la technologie d'accès utilisée (convergence fixe/mobile, déc. 2005)
- **JAIN** : Java APIs for Integrated Networks, 1998
  - Ensemble d'interfaces permettant de développer rapidement de nouveaux services de télécom. indépendamment du matériel utilisé
- **JSLEE** : JAIN Service Logic Execution Environment, 2003.
  - Portabilité des services, indépendance au réseau, ouverture…
- **SIP Services Architecture** : framework SAMM, Bell Labs 2002.
  - Réflexion sur la place à attribuer aux services dans les architectures

Revue de la littérature

- **Aspects retenus des modèles d'architecture étudiés :**
  - Modélisation en couches logiques
  - Mise en place de niveaux d'abstraction
  - Utilisation des standards pour l'interopérabilité
  - Mécanismes d'intégration de services
  - Composants chargés de la médiation des services

Revue de la littérature

- **Solutions basées sur des langages généralistes :**
  - Scripts CGI (Common Gateway Interface)
  - Servlets Java
  - Modules SIP Express Router, ...

  -> expressives mais peu sûres.

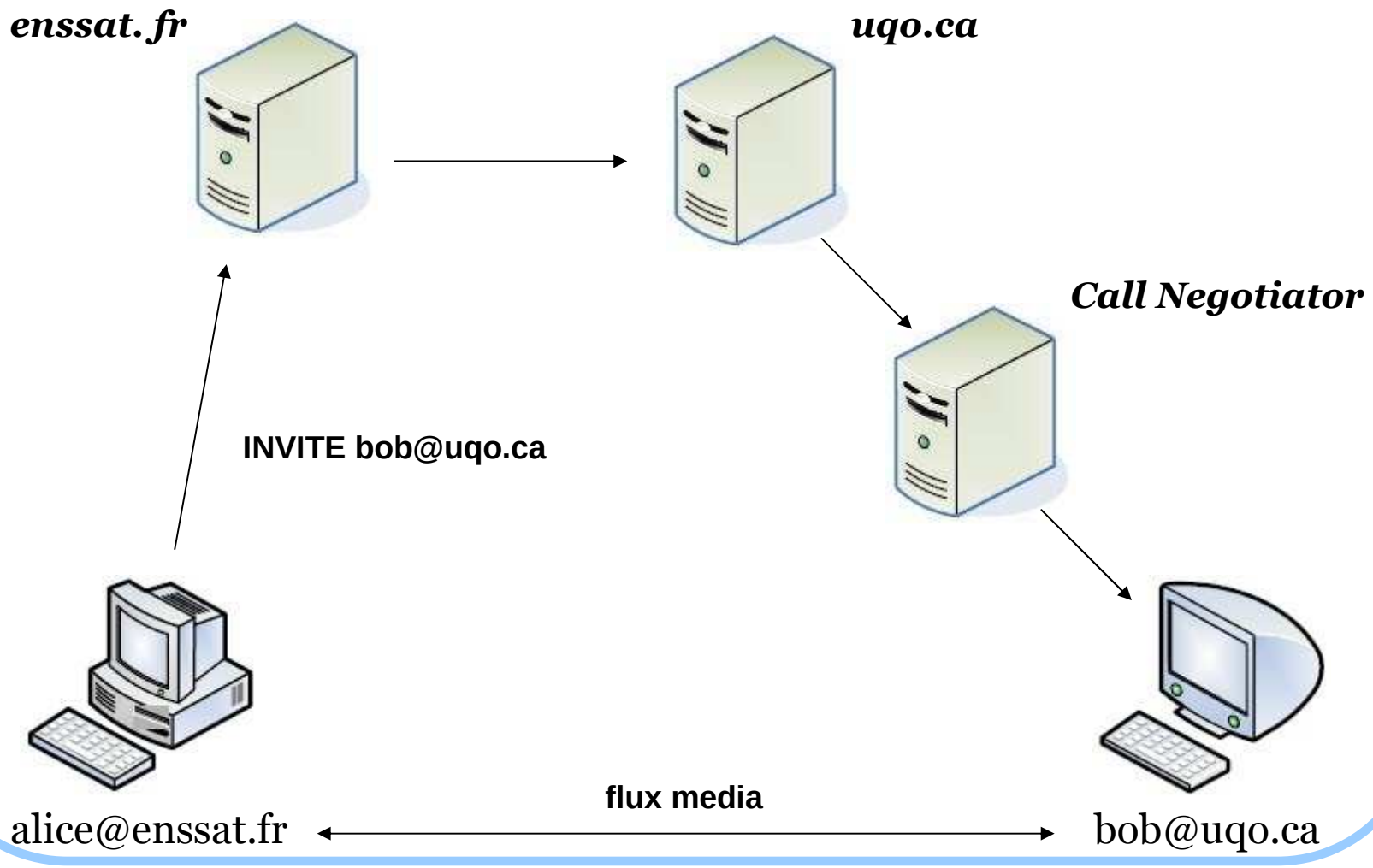
- **Solutions basées sur des langages restreints :**
  - CPL (Call Processing Language)
  - LESS (Language for End System Services)
  - MSPL (Microsoft SIP Processing Language), ...

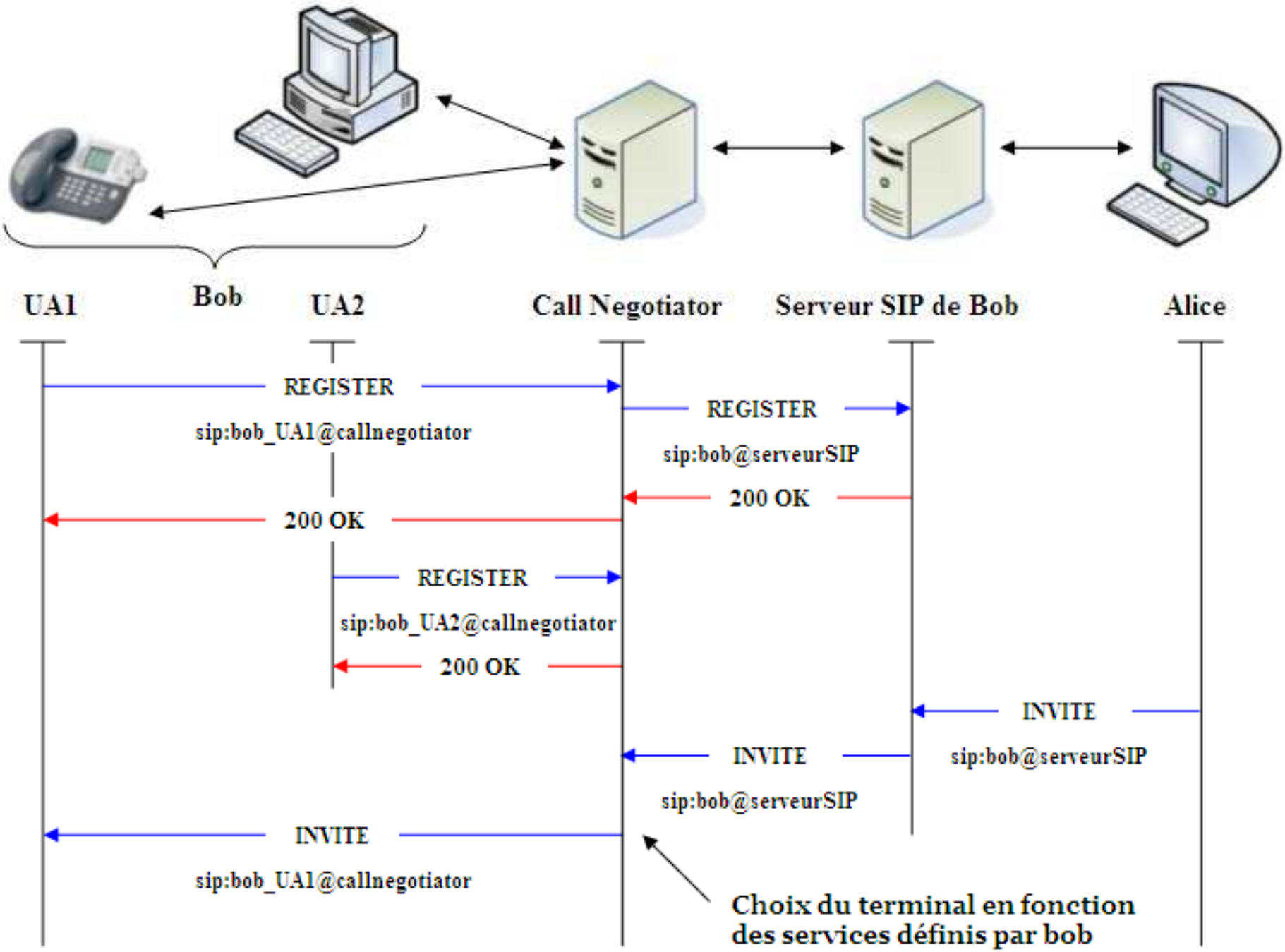
  -> sécuritaires mais limitées.

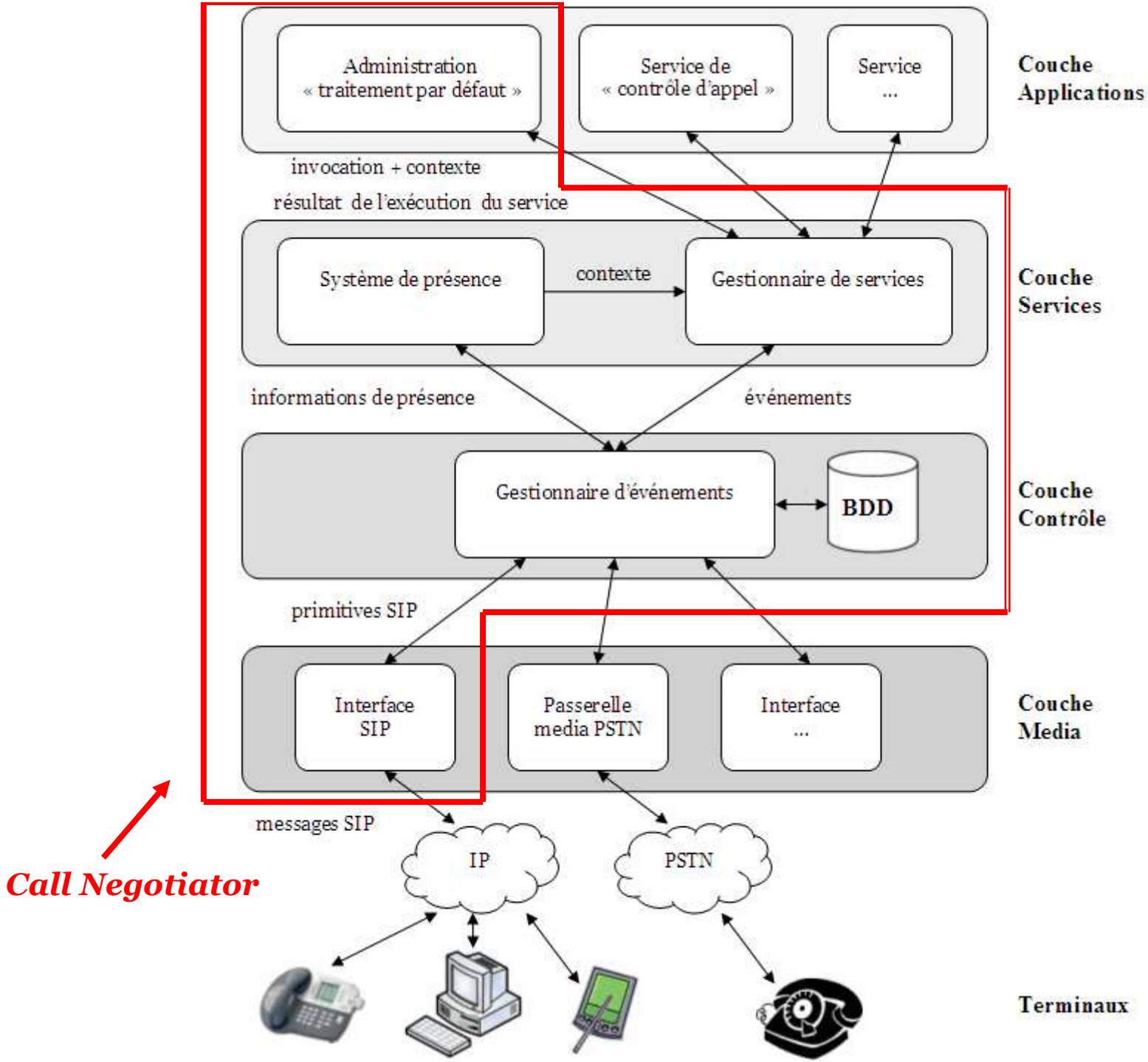
Revue de la littérature

- **Aspects retenus des systèmes et langages de programmation de services étudiés :**
  - Programmation à 2 niveaux
  - Mise en place de niveaux d'abstraction
  - Application de restrictions au niveau du langage
  - Mécanismes de vérification des services
  - Haut niveau de performance car fortement sollicité

Architecture de services




UA1
Bob
UA2
Call Negotiator
Serveur SIP de Bob
Alice
REGISTER
sip:bob_UA1@callnegotiator
REGISTER
sip:bob@serveurSIP
200 OK
200 OK
REGISTER
sip:bob_UA2@callnegotiator
200 OK
INVITE
sip:bob@serveurSIP
INVITE
sip:bob@serveurSIP
INVITE
sip:bob_UA1@callnegotiator
Choix du terminal en fonction
des services définis par bob


Administration
« traitement par défaut »
Service de
« contrôle d'appel »
Service
...
Couche
Applications
invocation + contexte
résultat de l'exécution du service
Système de présence
contexte
Gestionnaire de services
Couche
Services
informations de présence
événements
Gestionnaire d'événements
BDD
Couche
Contrôle
primitives SIP
Interface
SIP
Passerelle
media PSTN
Interface
...
Couche
Media
messages SIP
IP
PSTN
Terminaux
Call Negotiator

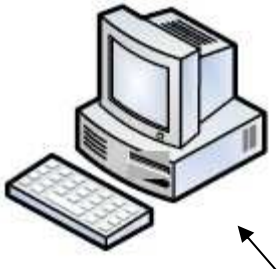
Avantages de ce modèle

- Indépendance vis-à-vis du serveur de communication
- Indépendance vis-à-vis du protocole de signalisation
- Respect des standards établis

  -> **plus grande interopérabilité !**

- Gestion des services facilitée
- Meilleur contrôle sur ses informations
- Mobilité accrue : indépendance au terminal
- Une grande variété de services possibles

Programmation de services



Interface web

si je reçois un appel provenant de john.doe@uqo.ca **alors** transférer l'appel à ma messagerie vocale.

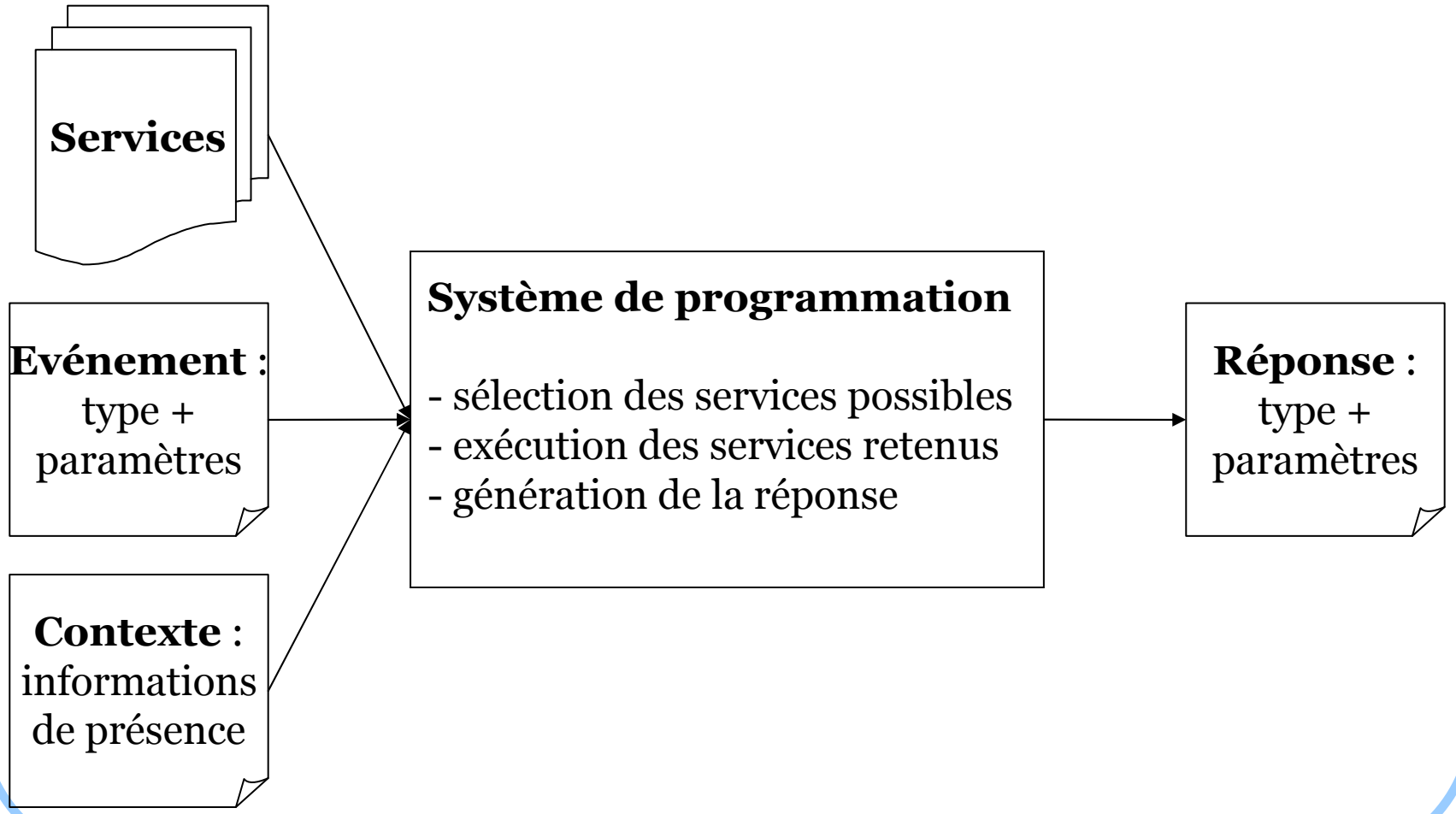
Terminaux SIP

Call Negotiator

Serveur SIP

Messages SIP

Système de simulation

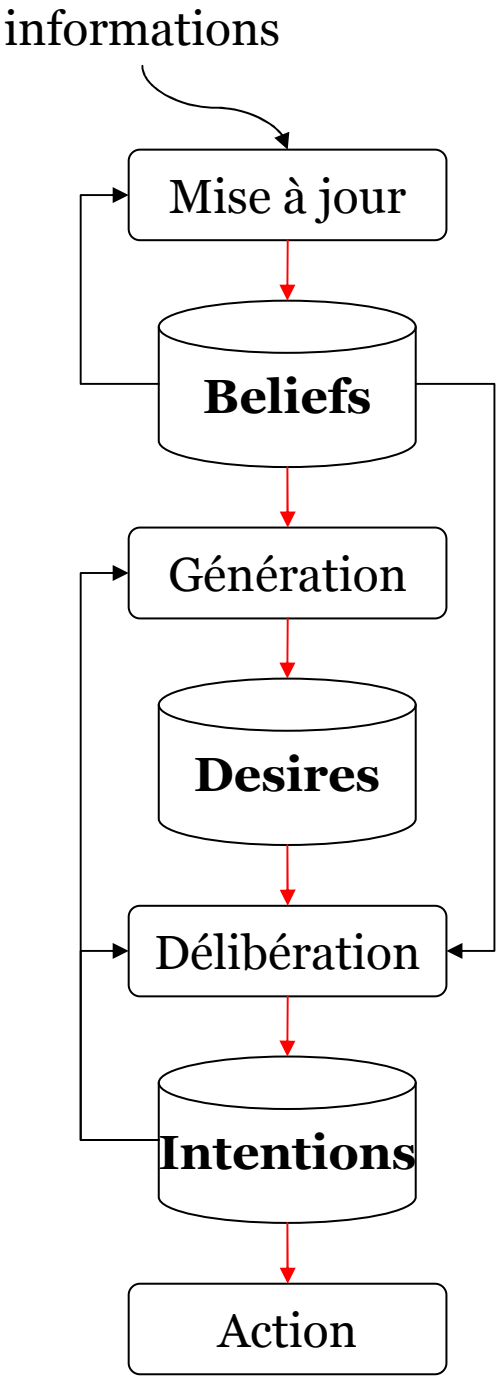


Modèle BDI

- **BDI** (*Beliefs, Desires, Intentions*) est un modèle cognitif à agents emprunté du domaine de l'intelligence artificielle.

- **Deux processus :**
  - décider quels buts poursuivre
  - décider comment les réaliser

- **Trois ensembles :**
  - *Beliefs* : ensemble des « croyances », ce que connaît l'agent. En ce qui nous concerne : les informations de présence, le contexte.
  - *Desires* : « désirs » ou « options », qui représentent l'ensemble des opportunités offertes à l'agent. Ici, les services programmés.
  - *Intentions* : les options retenues par l'agent, qui mènent aux actions. Dans notre cas : les services choisis, à exécuter.



(source: wikipedia)

Modèle BDI

Principe d'un système BDI :

1. collecte des informations

2. mise à jour de la base de « croyances » (contexte)

3. génération des options possibles, parmi l'ensemble des « désirs » (services).

4. délibération : choix et ordonnancement, en fonction des options retenues.

5. action : exécution des services.

Avantages :

- Sensibilité au contexte
- Sélection des plans à exécuter
- Récupération d'erreurs

Programmation de services

AgentSpeak (L) : langage permettant de programmer des agents BDI.

1. Définir un ensemble de croyances de base
2. plan : : = événement(s) déclencheur(s) : contexte <- action(s)

Ex: alice programme le service de transfert d'appel suivant :

- « si mon statut est occupé, transférer tous mes appels vers bob »

1. Base de connaissances

utilisateur(alice).
utilisateur(bob).
statut(alice, occupé).

2. Expression du plan

+invite(x,y) :
statut(y,occupé)
<- !transférer_appel(x,y,bob)

Conclusion

- **Travail accompli :**
  - Revue de la littérature
  - Elaboration d'un modèle d'architecture
- **Travaux prévus :**
  - Approfondissement de l'architecture
  - Conception et implémentation du système de programmation de services (système de simulation)
- **Travaux futurs :**
  - Interactions de fonctionnalités
  - Contexte : acquisition, format des données et cohérence