Journée RIS *Intergiciel et Sûreté de Fonctionnement*

6 juin 2002

Sûreté de Fonctionnement d'intergiciel CORBA: caractérisation de services par injection de faute

Eric Marsden

<emarsden@laas.fr>

Groupe TSF
LAAS-CNRS, Toulouse

Objectifs

Problématique COTS pour les intégrateurs d'intergiciel : besoin de caractérisation

- développer une méthodologie d'étalonnage de la robustesse d'un intergiciel
- obtenir des mesures permettant de comparer différentes implémentations entre elles
- développer des mécanismes d'encapsulation qui améliorent la robustesse du logiciel
- évaluer l'efficacité de ces mécanismes

► Approche expérimentale basée sur l'injection de faute

Injection de faute

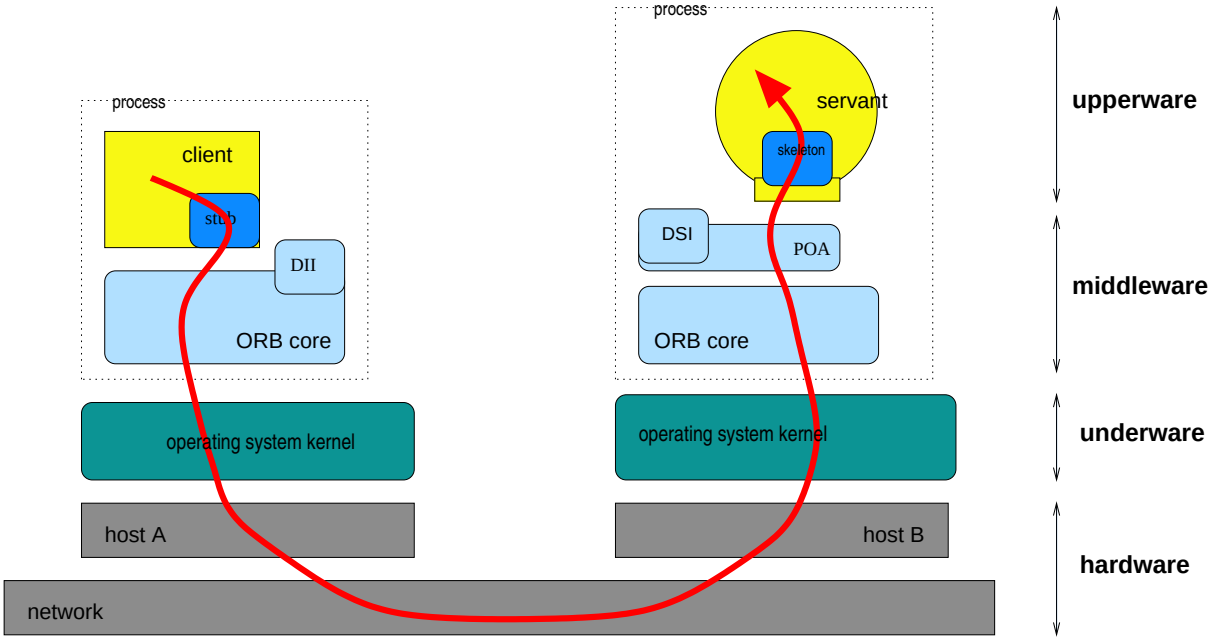
Méthode expérimentale de caractérisation de la SdF:

- examiner les réactions du système à des conditions anormales
- exercer ses mécanismes de detection d'erreur et de recouvrement
- obtenir des informations sur les modes de défaillance
- travaux précédents sur les micronoyaux

Éléments d'une campagne d'injection de faute:

- quelles fautes injecter, ou, et quand (représentatitivé)?
- quoi observer, et comment?
- quelle charge opérationelle (workload)?

Où injecter les fautes?



Quelles classes de faute?

Simuler des fautes physiques transitoires :

- corrompre le traffic réseau
- bitflip dans l'espace d'addressage du courtier

Simuler des fautes du logiciel :

- corrompre les paramètres d'appels à l'interface du courtier
- mutation de code dans la bibliothèque, stub/skel, interfaces IDL

Simuler des fautes «environnementales» :

- problèmes de ressource : manque de mémoire, coupure de connexions IIOP
- tenue en charge : mesurer la dégradation de la performance en fonction du nombre et de la complexité des requêtes

Cible expérimentale: COServices

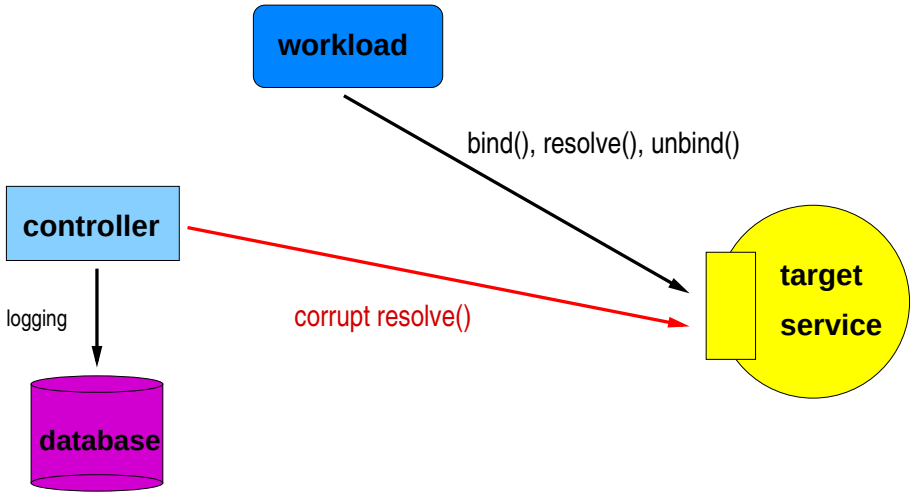
Premiers travaux sur les services de noms et d'évènement CORBA :

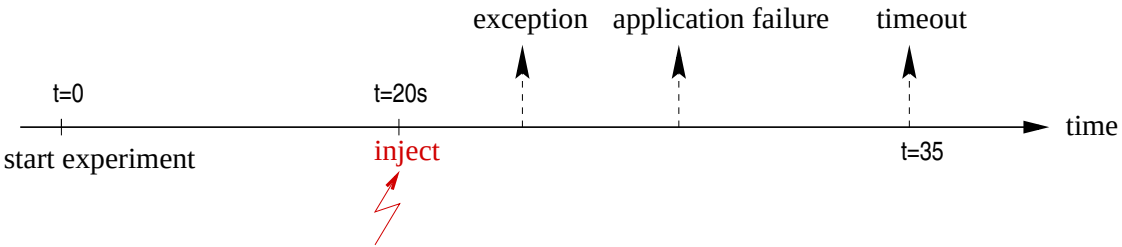
- interface standard : facile de comparer différentes implémentations
- point de défaillance unique dans un système à base d'intergiciel
- pas vers la caractérisation d'un courtier

Classification des modes de défaillance :

- crash du noyau
- crash du service
- hang du service : timeout sur une requête
- défaillance de l'application : l'erreur se propage à la charge
- exception CORBA

Configuration expérimentale





Modèle de fautes

Bitflips et double-zéro dans les requêtes IIOP entrantes, avec trigger temporel. On simule

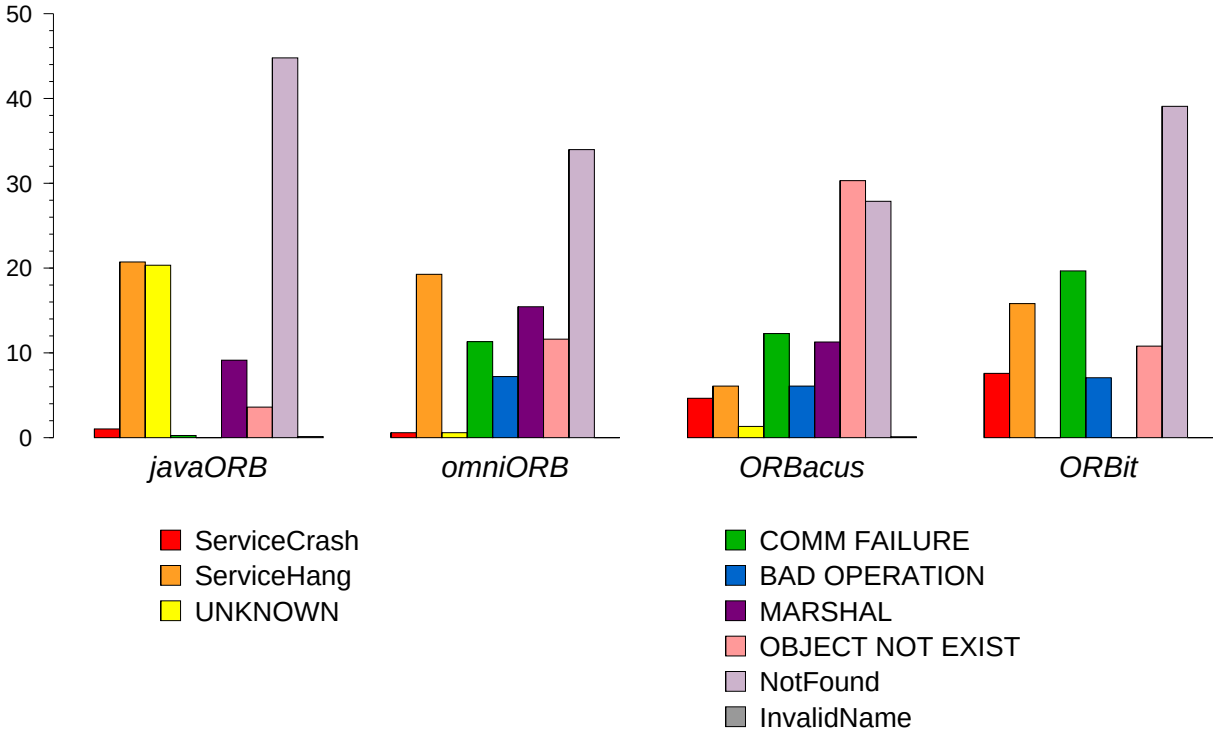
- fautes transitoires du système de communication
- propagation d'erreurs depuis les courtiers distants
- interactions avec des clients malicieux (déni de service)

Un message peut être corrompu à différents niveaux :

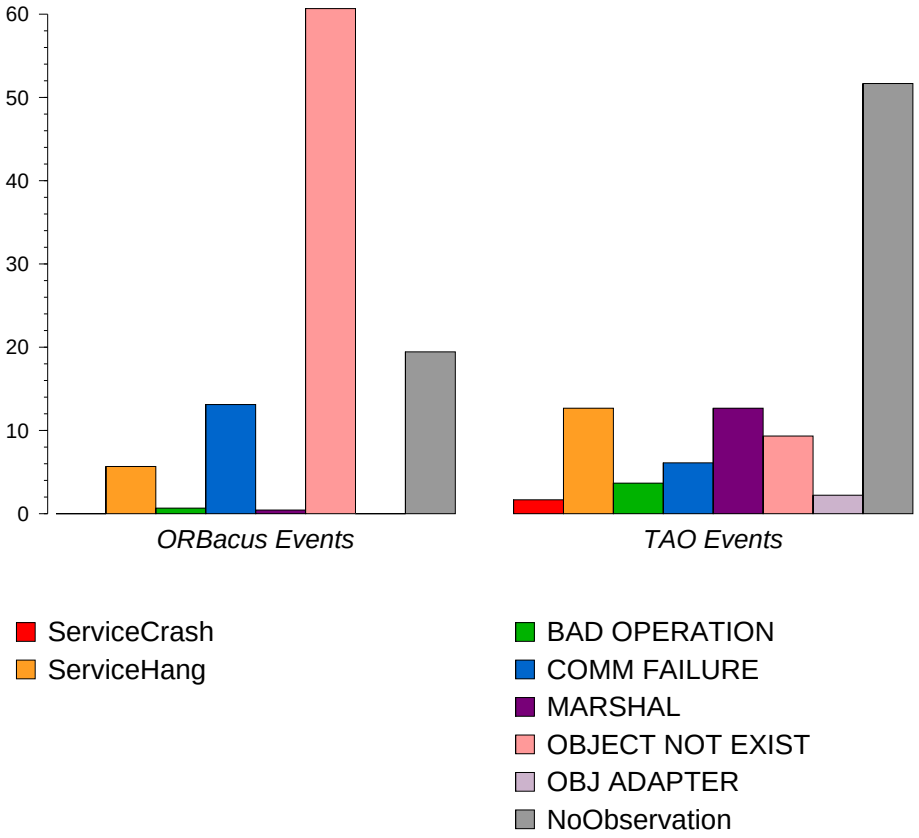
- *couche transport* : devrait être détecté par la pile IP
- *niveau applicatif* : dans les données encapsulées

Représentativité: relevé d'erreurs de communication [Stone & Partridge 2000]

Résultats NS: modèle de fautes bitflip

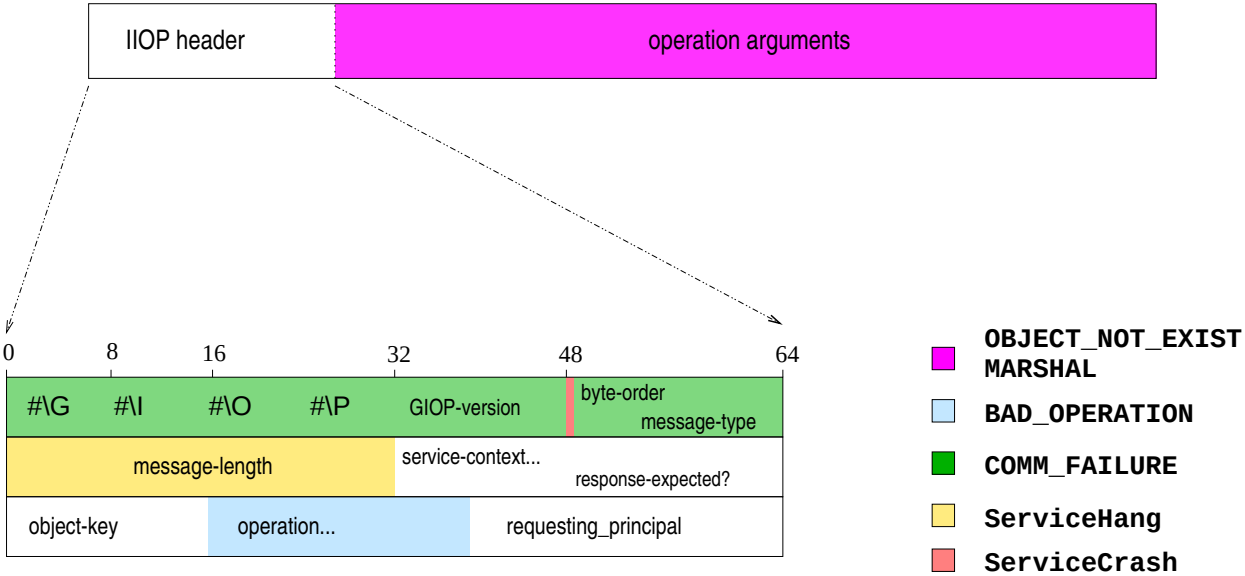


Résultats ES: modèle de fautes bitflip



Résultats NS: par position

Analyse des modes de défaillance suivant la position de la corruption dans le message :



Conclusions

- méthode d'analyse des modes de défaillance d'un service CORBA, par injection de messages IIOP corrompus
  - met en évidence la différence entre spéc et implémentation
  - technique portable : facile d'ajouter une nouvelle cible
  - méthode non intrusive
  - modèle de faute limité

- Ajouter un checksum niveau applicatif à GIOP







Perspectives

- étudier l'impact du système d'exploitation sous-jacent sur la robustesse
- effet de la charge réseau et machine sur les modes de défaillance
- améliorer les observations : mesurer la latence de détection d'erreur
- développer des mécanismes d'encapsulation pour les fautes identifiées
- travailler sur d'autres cibles et d'autres modèles de faute :
  - ◊ bitflips en mémoire
  - ◊ erreurs de ressources
  - ◊ tenue en charge
  - ◊ mutation de code

Représentativité des fautes

Article de Stone et Partridge, SIGCOMM2000 :

- un segment TCP corrompu sur $\approx$ 30000
- checksum TCP de 16 bits complément à un
- taux d'erreur résiduel (non détéctés) de 1 par 3.3TB
- soit toutes les 80 heures sur un LAN 100MB/s saturé

Sources des erreurs :

- matériel : cartes réseau, routeurs
- logiciel : piles de protocole des noyaux

▶ TCP est supposé fiable, mais ne l'est pas à 100% !

Résultats: autres observations

- injections dans le champ longueur de la trame IIOP provoquent des timeouts
- bitflips à des positions multiples de 8 sont souvent non détectées
- les bits de poids fort du byte order sont ignorés. La corruption des bits de poids faible provoque le crash du service.
- dans certains cas, *plusieurs* conditions sont observées (*e.g.*, défaillance de l'application couplé à une exception CORBA)
- les expériences sont essentiellement déterministes, malgré la présence de non-déterminisme dans le noyau et le réseau

Travaux antérieurs

Validation par injection de faute :

- Ballista et corruption des paramètres d'appels au courtier coté client (Koopman, DSN2001)
- corruptions réseau ciblant un intergiciel spécifique sur CAN (Koopman, FTCS'98)
- *ORCHESTRA*: test de protocole par injection de faute (Jahanian, University of Michigan)
- l'outil *N-FTAPE* : injection de faute dans les réseaux (CRHC, UIUC)

Validation par test fonctionnel :

- *CORVAL*: tests de conformance du *Open Group*, et le projet *CORVAL2*

Réception de données inattendues

- Simule la présence de clients défaillants ou malveillants
  - envoyer des messages IIOP de type inattendu (*eg* `CancelRequest` pour un request-id bidon)
  - envoyer des messages avec des exceptions inattendues
  - envoyer des messages ayant des service context bizarres
  - négociation bizarre lors de l'établissement de la connexion IIOP







Fautes de ressource

- Simule des fautes provenant de l'environnement d'exécution
  - coupure de session IIOP
  - manque de mémoire
  - disque local rempli
  - charge CPU élevée
  - défaillance de services CORBA

On mesure l'étanchieté de l'intergiciel vis-à-vis de défaillances du support d'exécution. Si le courtier est robuste, il signalera ces conditions à l'application, par exemple en soulevant des exceptions.

Mutation de code

Simulation de fautes du logiciel. Les mutations peuvent être introduites à différents niveaux :

- code applicatif
- code généré par le compilateur IDL
- bibliothèque ORB
- dans l'interface IDL