FM&&T

Model Checking Publish/Subscribe Event Notification

Maurice ter Beek Stefania Gnesi

Mieke Massink Diego Latella

FM&&T, ISTI–CNR, Pisa, Italy

Outline

- Publish/subscribe event notification
- Running example: file storage and retrieval
- Traditional model checking with Spin and LTL
- Security analysis of (tele)communication protocols

Publish/subscribe event notification

Raise user awareness by intelligent data sharing:

> *whenever a user publishes a document, automatically all users that are subscribed to that document are notified via an asynchronous multicast communication*

This notion is nowadays much studied in the groupware literature:

- \+ "full decoupling of the communicating participants in time, space and flow" [EFGK03]
- – generally difficult to verify [GKK03,ZGB03]

Our aim: formally model and verify the addition of such a service

Example: file storage and retrieval

Minimal set of database operations:

get – extract a read-only copy of a file

import – insert a new file

checkout – extract an exclusive copy of a file

checkin – replace an edited (previously checked out) file

Additional publish/subscribe operations:

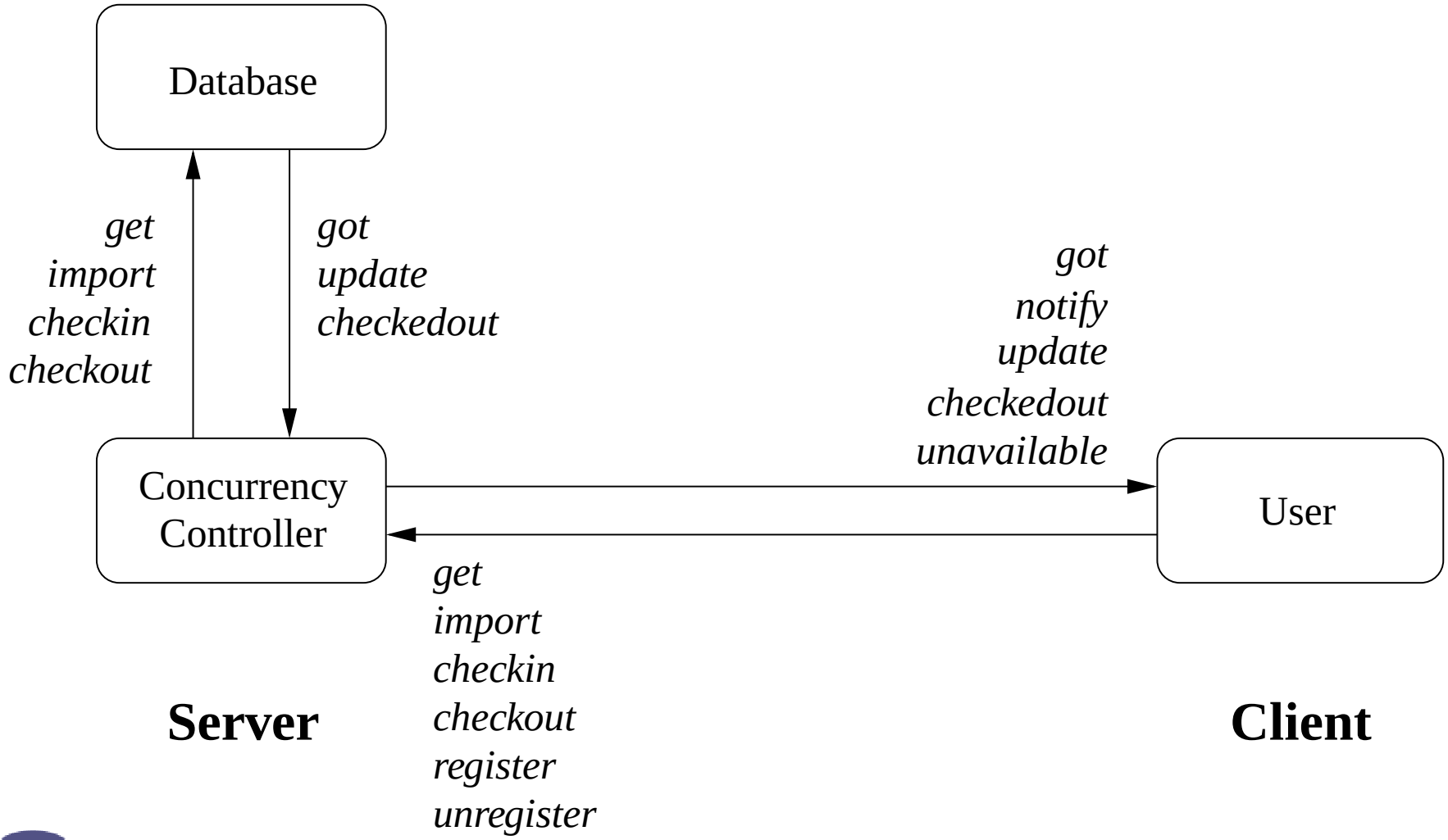
user can explicitly *(un)subscribe* to a file (implicitly via *get*!)

Every user subscribed to a file receives:

notify – when the file is checked out by another user

update – when another user returns the file to the database

Example publish/subscribe protocol



Traditional model checking

An automatic technique to verify whether a concurrent system design satisfies its specifications

Very hard in standard ways (like testing) due to non-determinism and interleaving

+ does exhaustive verification: takes into account all possible input combinations and states

– risk of running out of memory due to a state-space explosion

⇒ a simplified model is used, still capturing the core of the system design while abstracting from unnecessary details

Spin and Promela

Spin — Simple Promela INterpreter:

- a state-of-the-art and on-the-fly model checker
- developed at Bell Labs by G.J. Holzmann
- to formally verify distributed systems specified in Promela
- able to verify deadlocks / assertions / unreachable code / LTL formulae / liveness / etc.
- always provides a counterexample if a property is violated

Promela — PROcess MEta LAnguage:

- a non-deterministic C-like specification language
- borrows notation for I/O operations from process algebra CSP
- results in finite-state systems communicating by channels

Example: excerpt from User process

```
proctype User(byte id) {
  byte edit[numFiles], registered[numFiles];
  bool waitingForCheckedOut = false;
  do
  :: (!waitingForCheckedOut) ->
       userToCC!get,id;
       doneGet: skip;
       ccToUser[id]?got;
       registered[0] = true

  :: ...

  :: (!edit[0] && !waitingForCheckedOut) ->
       waitingForCheckedOut = true;
       userToCC!checkOut,id
  od }
```

Verifying properties with LTL

LTL — Linear Temporal Logic:

- a propositional logic with temporal operators
- heavily used for specifying liveness properties
- property must hold always / eventually / until / etc.

Example: user can always eventually *get* a file

In LTL: $[\,]\ <>$ User[pid]@doneGet

id User process — label

$\forall$ trace $\forall$ state: *get* x

Spin: not valid, so a counterexample (MSC) is provided (< 1 sec.)

Denial of Service

Denial of Service (DoS): “no user is forever denied a service”

⇒ traditional model checking with Spin shows DoS may occur in given protocol [BMLGFS04]

Reason: “one user can endlessly keep the CC process busy”

⇒ unavoidable property of protocol, due to file access based on ‘retrial’ principle (i.e. no queue or file reservation system)

Question: how to quantify this problem?

Answer: Stochastic model checking!

Analysis of telecom protocols

With Gabriele Lenzini of Telematica Instituut, The Netherlands

- continue our collaboration on formal security analysis of telecommunication protocols
  (as previously done for an OSA/Parlay Authentication Interface)
- this time focus also on QoS, privacy issues and constraints
- might need to extend existing formal models, such as automata, to allow for predicates associated to actions (in order to mimic the constraints attached to the messages sent) etc.
- not so obvious, however, which tool to use for such analyses