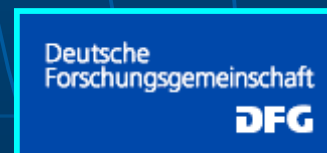


Unstoppable Stateful PHP Web Services

German Shegalov, Gerhard Weikum,
and Klaus Berberich

funded by



- Problem Statement and Background
- Interaction Contracts (IC) Framework
 - Contract between Web Services
 - Contract between User & Browser
- Implementation & Experiments:
Exactly-Once Web Service (EOS)
- Summary

Please review and place your order

Place your order

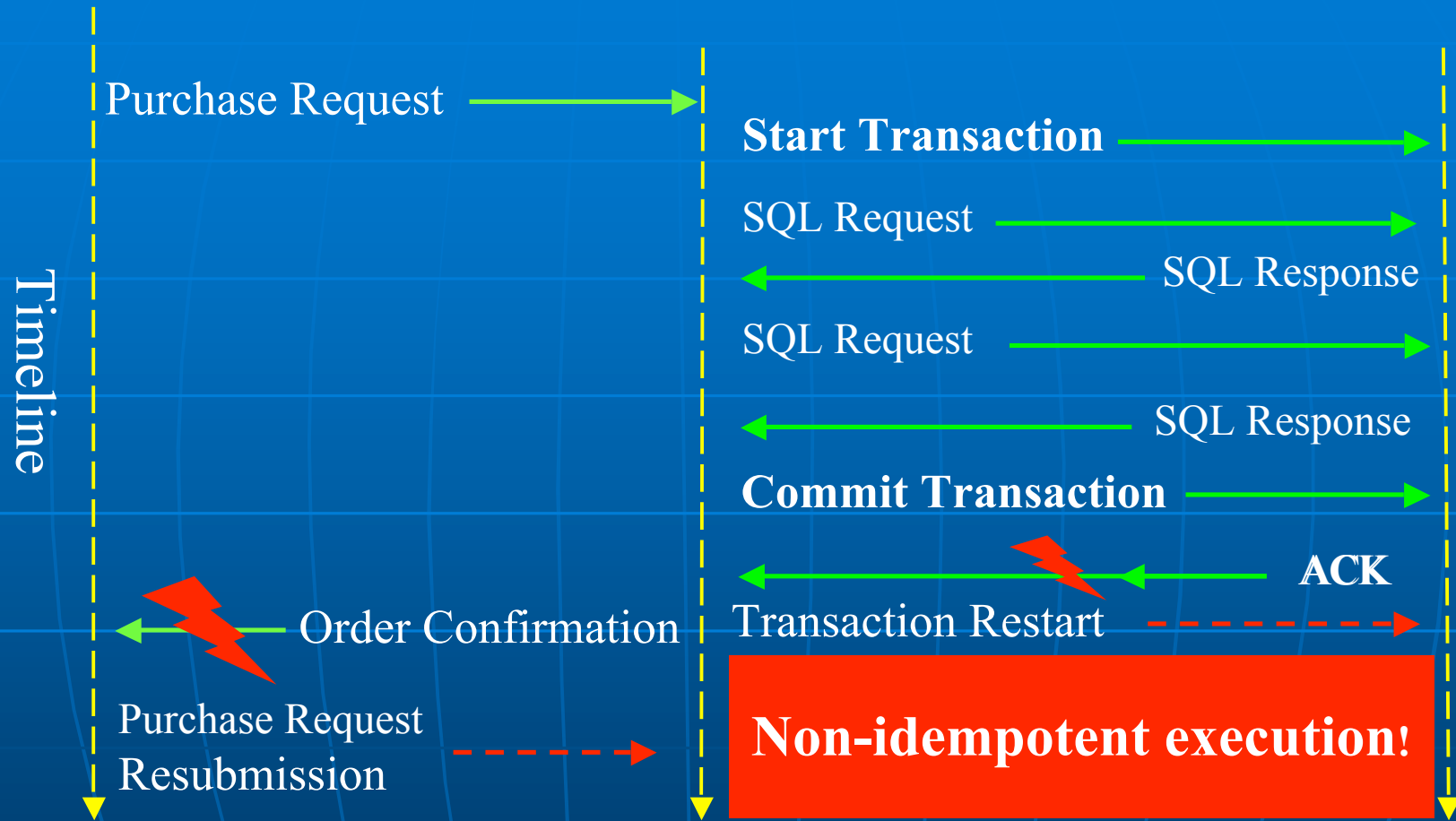
**Your server command (process id #20) has been terminated.
Re-run your command (severity 13) in
/export/home/WWW/your-reliable-eshop.biz/mb_1300_db.mb1**

Transactional Recovery Insufficient

Web Client

Web Application Server

Database Server



- Atomic = At-most-once \neq Exactly-once
- Idempotence needs testability, but testable state all but simple

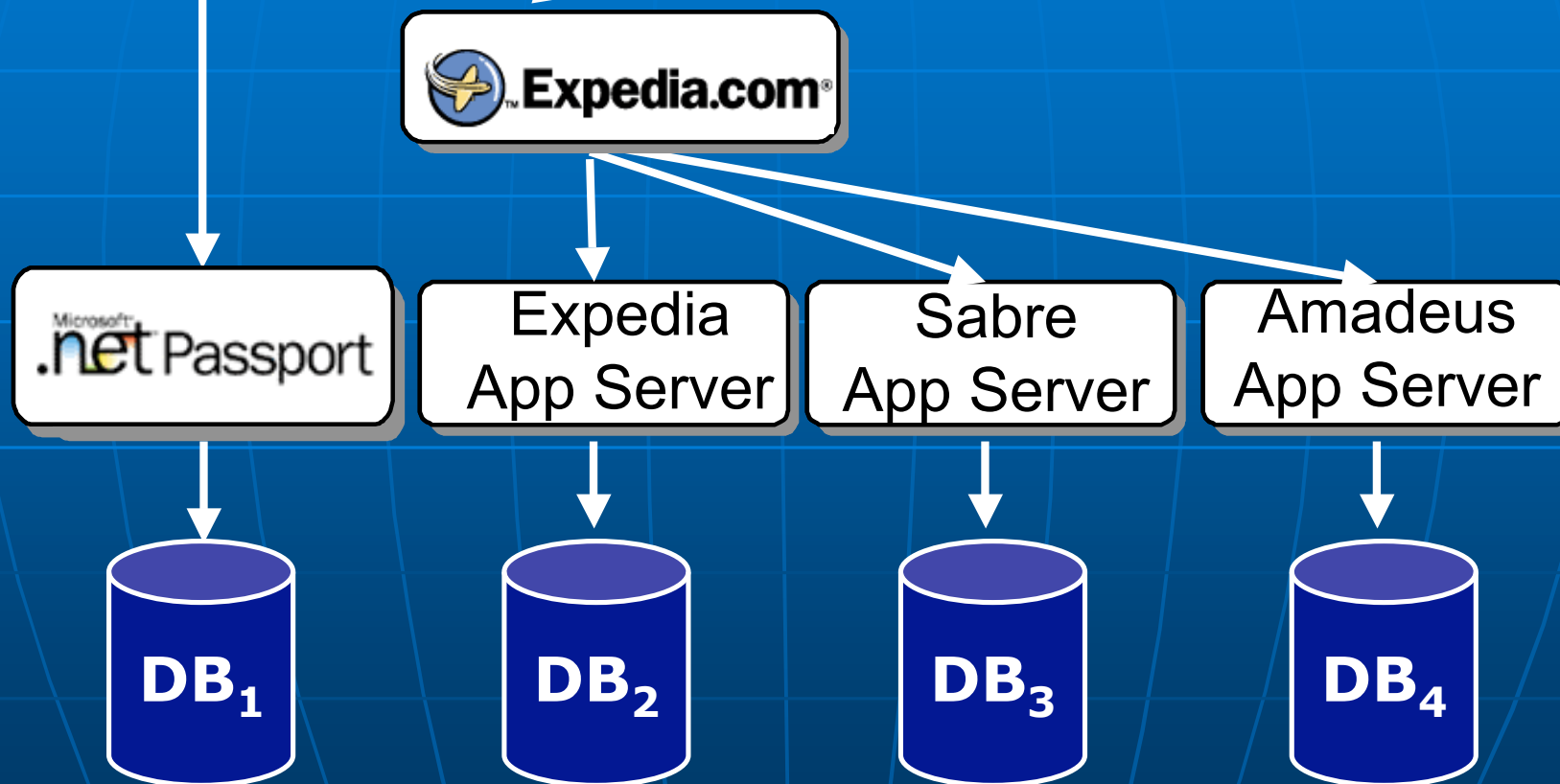
Real-World *n*-Tier App



Client

Complicated enough?

Business transactions between peers
in community apps (Skype, MSN, ...).



Problem Statement



- Application-tailored solutions for high-end e-business not available to masses of low-end service & mashup programmers
- Ease programming by providing generic solution in Web-service middleware
- Should mask all (but inevitably disruptive) failures to application program(mer)s: message, process, data failures

- Problem Statement and Background
- Interaction Contracts (IC) Framework
 - Contract between Web Services
 - Contract between User & Browser
- Implementation & Experiments:
Exactly-Once Web Service (EOS)
- Summary

- **Components and Guarantees**
 - **Persistent Pcom**: **Persistent, testable** state and messages
 - **External Xcom** (e.g., humans): ~~guarantees~~
- **Bilateral Interaction Contracts**
 - $Xcom \leftrightarrow Pcom = \text{External IC (XIC)}$
 - $Pcom \leftrightarrow Pcom = \text{Committed IC (CIC)}$
- **Composing IC's for Entire System: Exactly-Once Semantics**
 - App programs don't need to handle failures

- **Redo Log & Recovery Managers**
 - **Piecewise determinism** + Logging = Full Determinism
 - **Deterministic replay** recovers Pcom's
 - **Installation Points** speed up replay
 - Failure model
 - **Crashes**
 - **Message losses**
 - ~~Malicious manipulations~~
 - ~~Disk corruption~~ (sufficient redundancy)
- } Transient failures due to nondeterministic **Heisenbugs**

- Problem Statement and Background
- Interaction Contracts (IC) Framework
 - Contract between Web Services
 - Contract between User & Browser
- Implementation & Experiments:
Exactly-Once Web Service (EOS)
- Summary

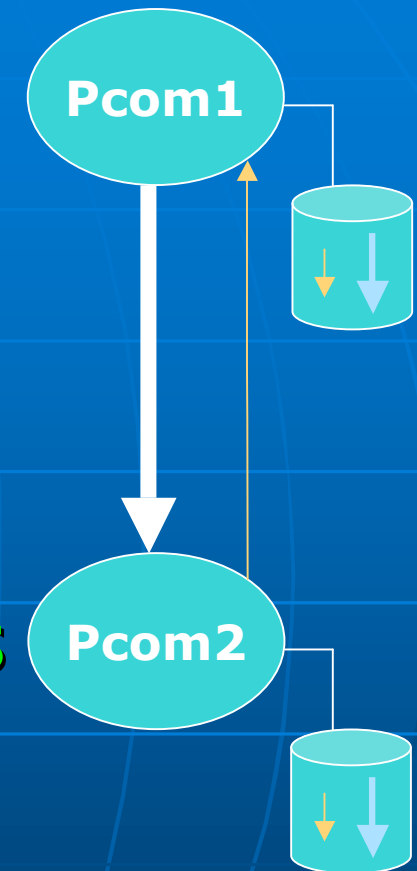
CIC Principles

■ CIC sender (Pcom1) obligations

- Persist state before send
- Tag message with a **MSN**
- Resend on timeout until **stable** ack
- Resend on receiver's inquiry
- Forget interaction on installed ack

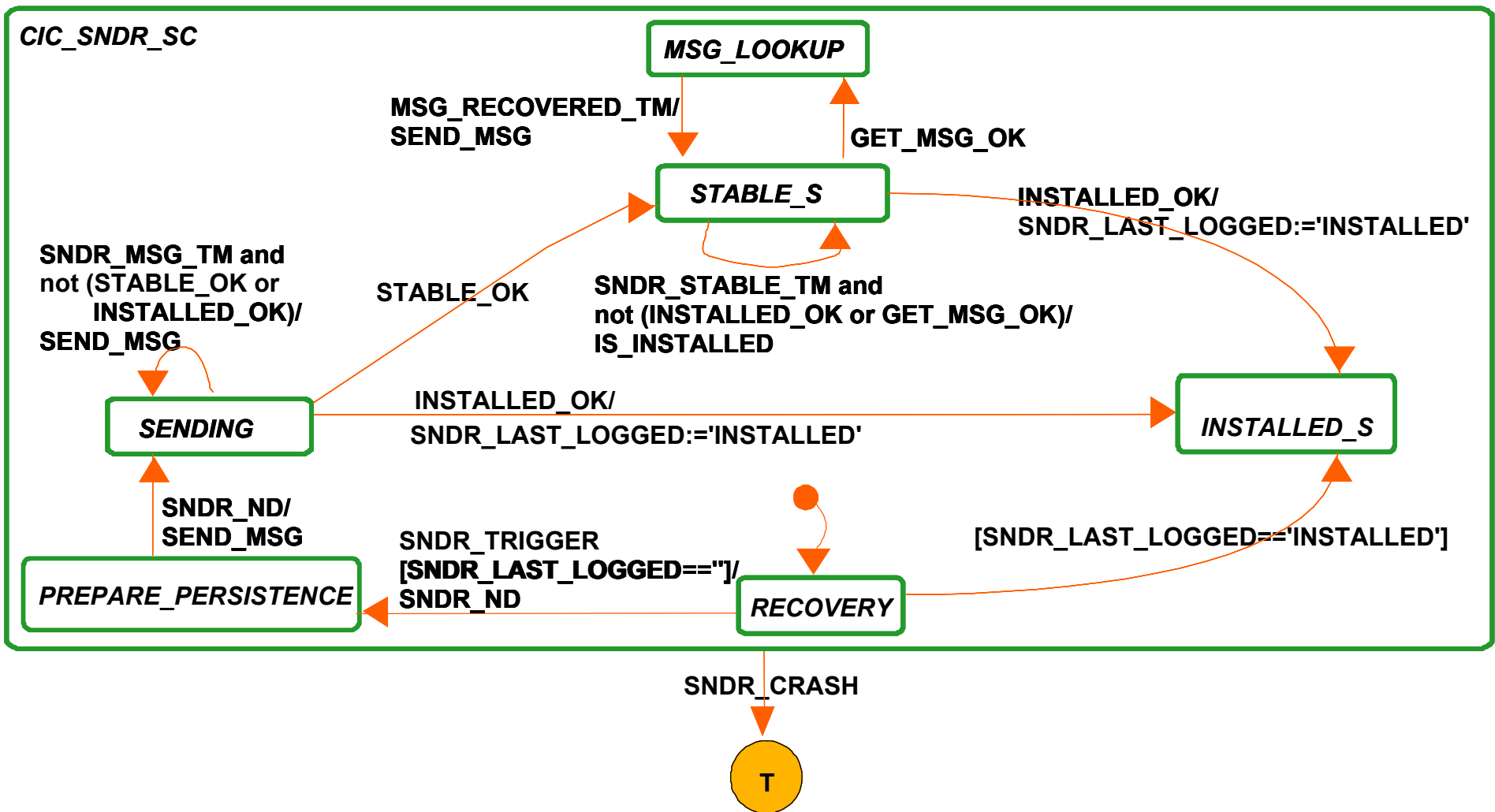
■ CIC receiver (Pcom2) obligations

- Eliminates duplicates by **MSN**
- Persists interaction before **stable** ack
- Inquires msg body if not in local log
- Ensures autonomous recovery before installed ack



Weaker than using persistent queue
or installing state for each interaction

Statechart for CIC Sender

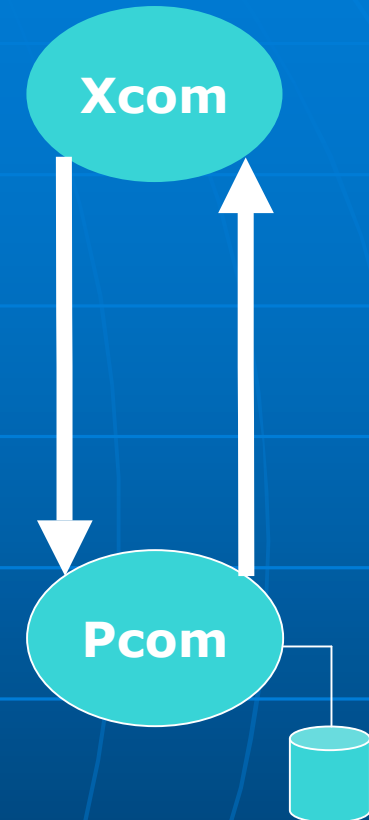


* **EVENT_OK** = **EVENT** \wedge \neg **LINK_OUTAGE** **_TM** means **TIMEOUT**

- Problem Statement and Background
- Interaction Contracts (IC) Framework
 - Contract between Web Services
 - Contract between User & Browser
- Implementation & Experiments:
Exactly-Once Web Service (EOS)
- Summary

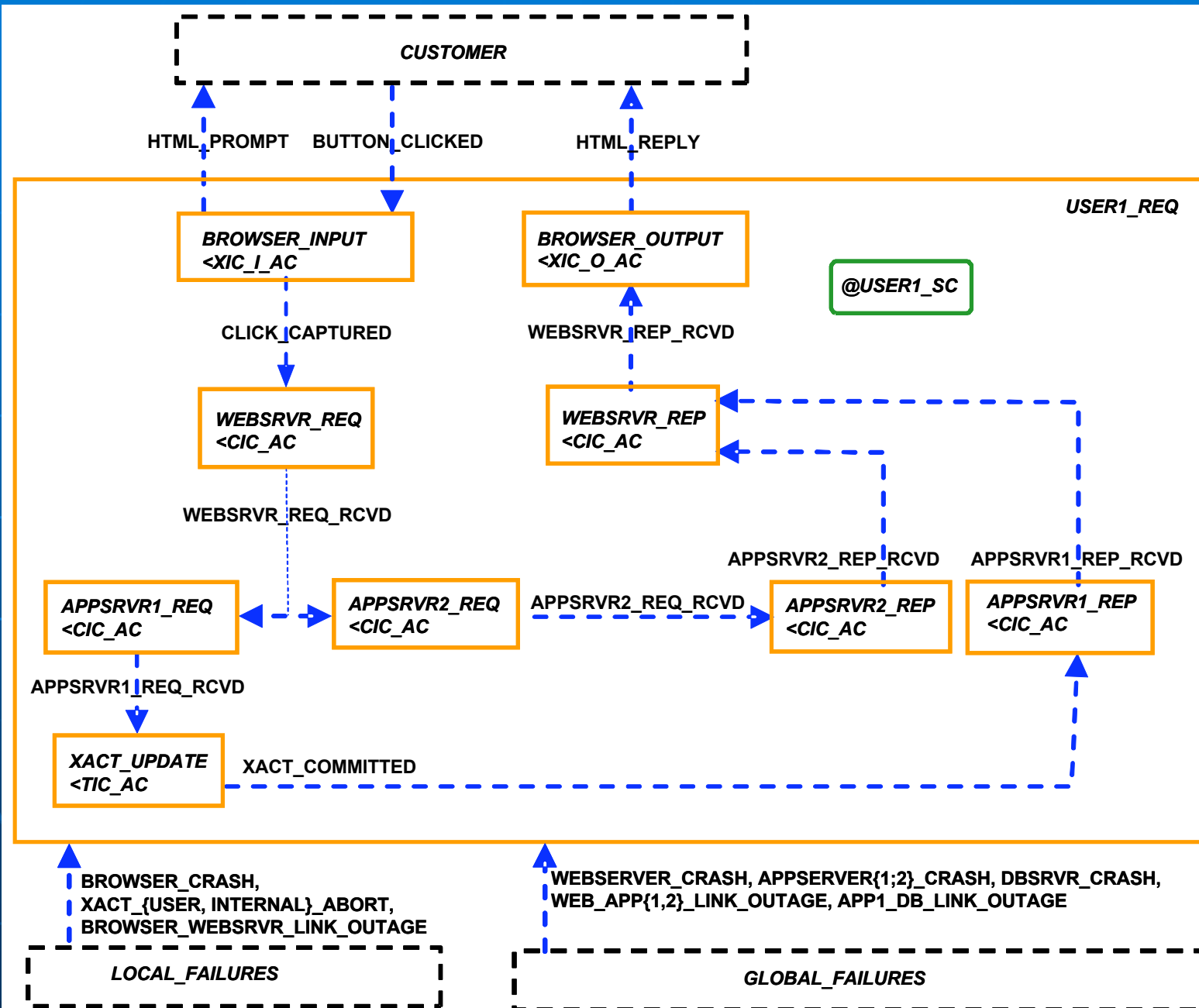
XIC Principles

- XIC sender Xcom obligations
 - None (but should resend on timeout)
- XIC receiver Pcom obligations
 - Persist interaction immediately
- XIC sender Pcom obligations
 - Persist state before sending message
 - Resend message after Pcom failure
- XIC receiver Xcom obligations
 - None



Typical setup: Xcom = user, Pcom = browser
Some failures inherently non-maskable

IC's for Composite Web Service



- Problem Statement and Background
- Interaction Contracts (IC) Framework
 - Contract between Web Services
 - Contract between User & Browser
- Implementation & Experiments:
Exactly-Once Web Service (EOS)
- Summary

EOS: Exactly Once Web Service

EOS prototype implementation:

Exactly-once semantics

for composite services with

- Transparent EOS-enabling of Web pages by piggybacked Javascript
- **XIC's: Transparent browser recovery** via callbacks and browser-specific XML store
- **CIC's: App server recovery** by modified session mgt. of Apache and Zend engine
- Fully transparent: **no changes to app code** neither to PHP scripts nor to browser

Efficiency: Message Lookup Tables

- Built at Pcom during normal operation
- Rebuilt from the log during Pcom recovery
- Input MLT for **duplicate elimination**
- Output MLT to track CIC progress enabling timely **garbage collection**

OMLT of eosphp3

URI	MSN	CIC status
<i>http://eosphp1/auctions/</i>	3	<i>installed</i>
<i>http://eosphp2/books/</i>	5	<i>stable</i>
<i>http://eosphp1/auctions/</i>	6	<i>unknown</i>
<i>http://eosphp1/auctions/</i>	7	<i>installed</i>
<i>http://eosphp2/books/</i>	8	<i>installed</i>

IMLT of eosphp1

client id	MSN	Reply LSN
eosphp3	3	324
...

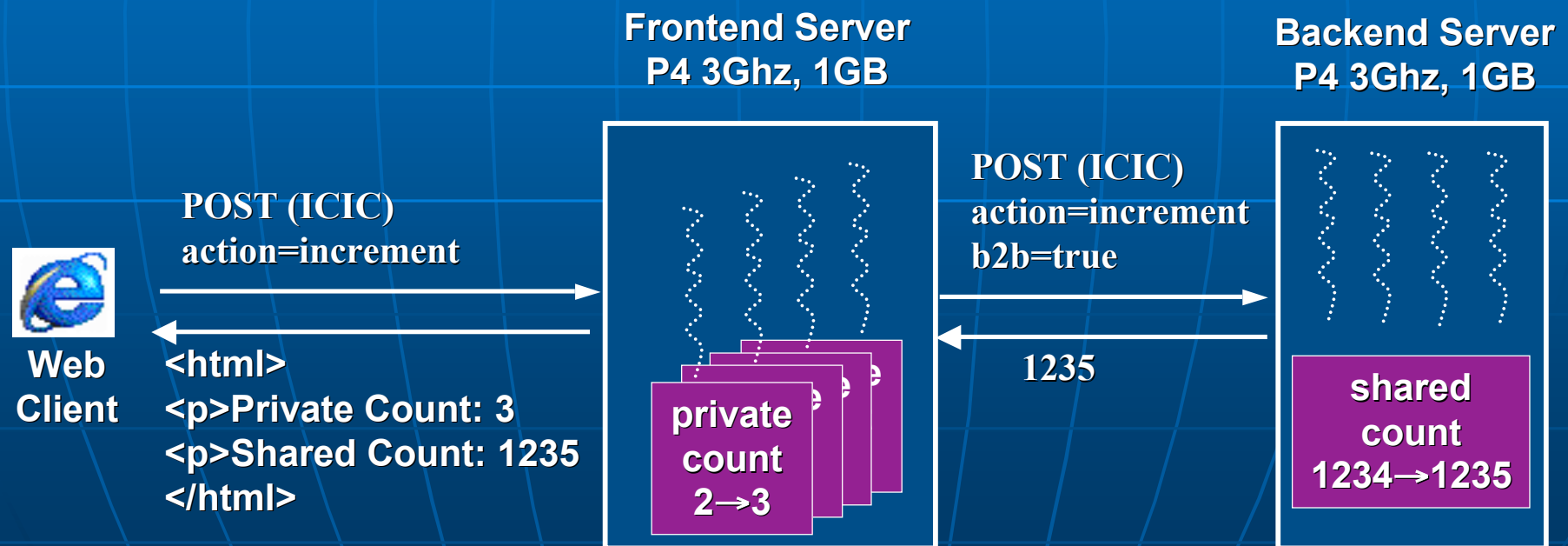
IMLT of eosphp2

client id	MSN	Reply LSN
eosphp3	5	324
...

Experiment Setup



- eBay-like auction service
- User settings at frontend (private)
- Auction items at backend (shared)
- 5 concurrent end users, synthetic load



PHP Scripts (not changed at all!)



```
<html>
  Script called 5 times
  Other server reports: Script called 1000 times
</html>
```

```
5.     printf("Script called %i times",
6.         $HTTP_SESSION_VARS["count"]);

7.     $ch = curl_init("http://eos-php.net/b2b.php");
8.     $b2b_reply = curl_exec($ch);
9.     printf("Other server reports: %s", $b2b_reply);
10.    curl_close($ch);
11. ?>
12. </html>
```

Zend Engine

Session | CURL

Zend Engine

Session | CURL

Run-Time Overhead



Session	1 step	5 steps	10 steps
PHP elapsed time [sec]	0.1560	0.7900	1.6100
EOS-PHP elapsed time [sec]	0.3140	1.6850	3.1000
Overhead (elapsed time) [%]	101%	113%	93%
PHP frontend CPU time [sec]	0.0390	0.2708	0.5727
EOS-PHP frontend CPU time [sec]	0.0815	0.6000	1.1545
Overhead (frontend CPU) [%]	109%	122%	102%
PHP backend CPU time [sec]	0.0090	0.0550	0.1200
EOS-PHP backend CPU time [sec]	0.0130	0.0750	0.1600
Overhead (backend CPU) [%]	44%	36%	33%

- Problem Statement and Background
- Interaction Contracts (IC) Framework
 - Contract between Web Services
 - Contract between User & Browser
- Implementation & Experiments:
Exactly-Once Web Service (EOS)
- Summary

Summary



- Generic IC's for composable services with exactly-once execution guarantee
- Eases programming by masking all (but inevitably non-maskable) failures
- Formal specification by state/activity-charts & model-checked CTL properties
- Efficient implementation:
EOS prototype for Apache/PHP

Thank you!