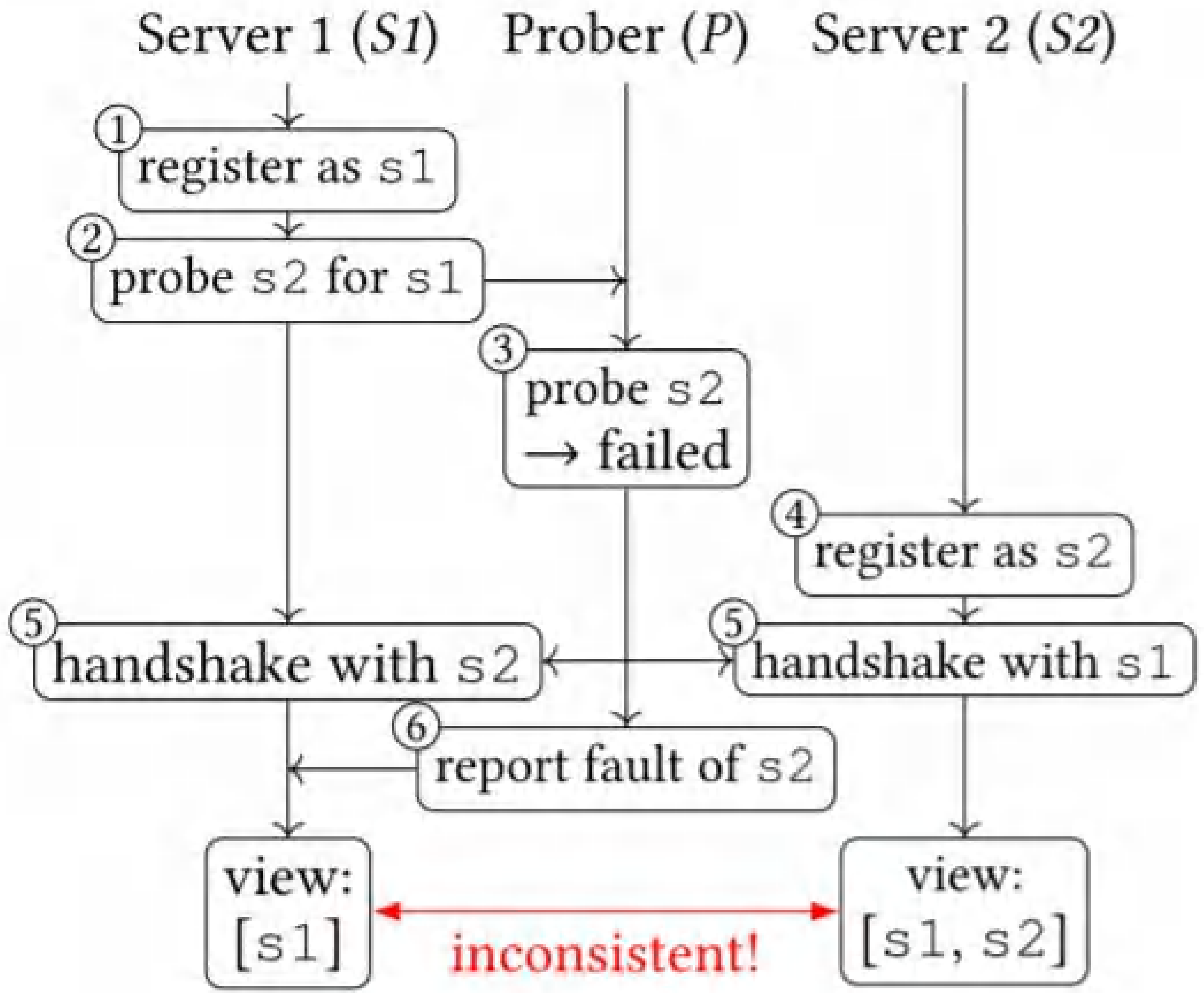
Effective Concurrency Testing for Distributed Systems

汇报人：张俊

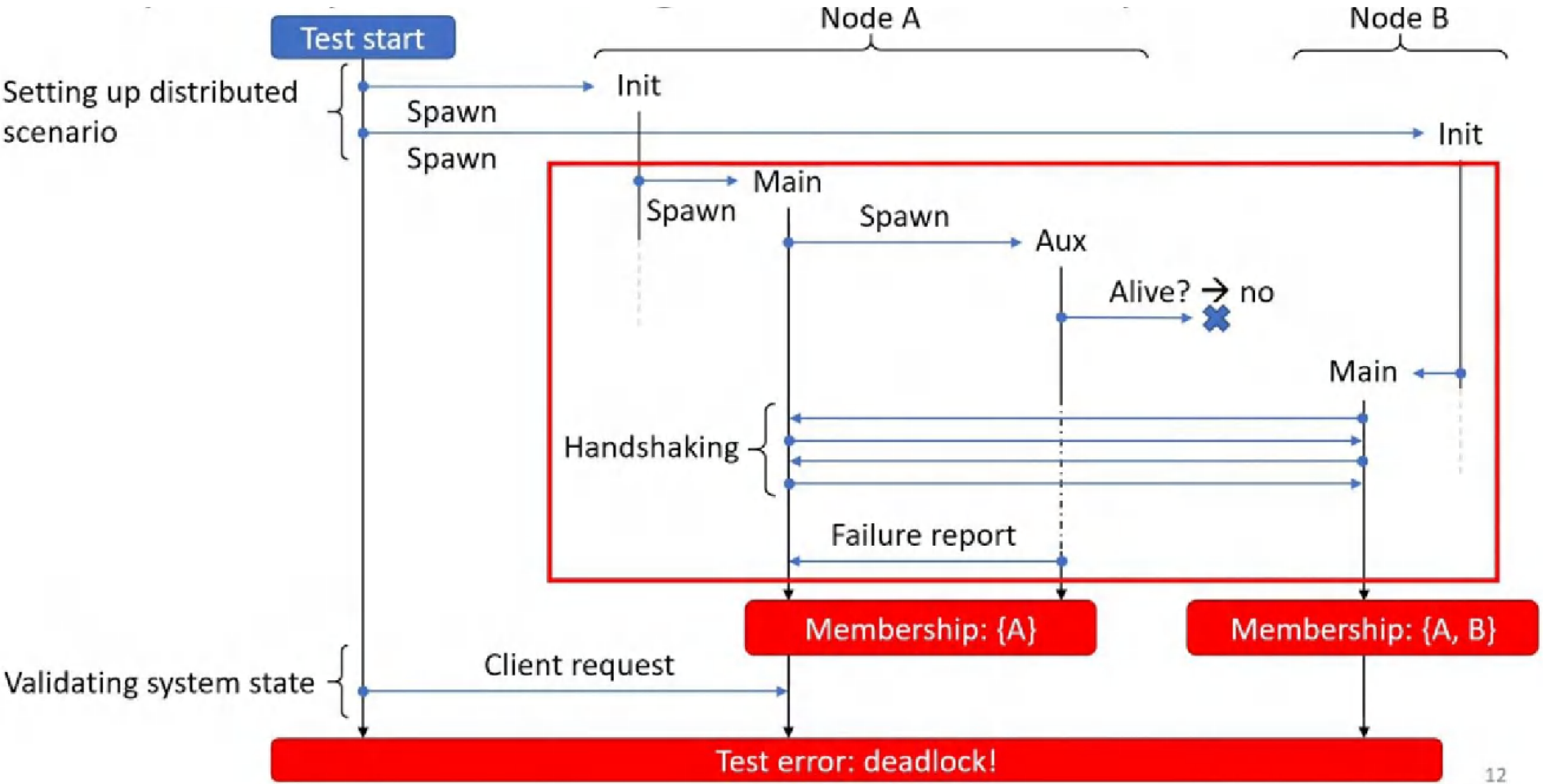
Concurrency is crucial but difficult

- Parallel nodes
- Untimely communication
- Concurrent requests
- Distributed failures


Server 1 (*S1*)
Prober (*P*)
Server 2 (*S2*)
1
register as s1
2
probe s2 for s1
3
probe s2
→ failed
4
register as s2
5
handshake with s2
5
handshake with s1
6
report fault of s2
view:
[s1]
inconsistent!
view:
[s1, s2]

- Systematic testing (model checking)
  - Exhaustively enumerate all possible interleavings
  - The interleaving space grows exponentially – quickly become intractable

- Randomized testing
  - Sample interleavings – simple and lightweight
  - Probabilistic concurrency testing (PCT) [1]
    - Coincide simple root causes by interleaving a small subset of ops
    - Still suffer from complex tests – too many operations to choose from!

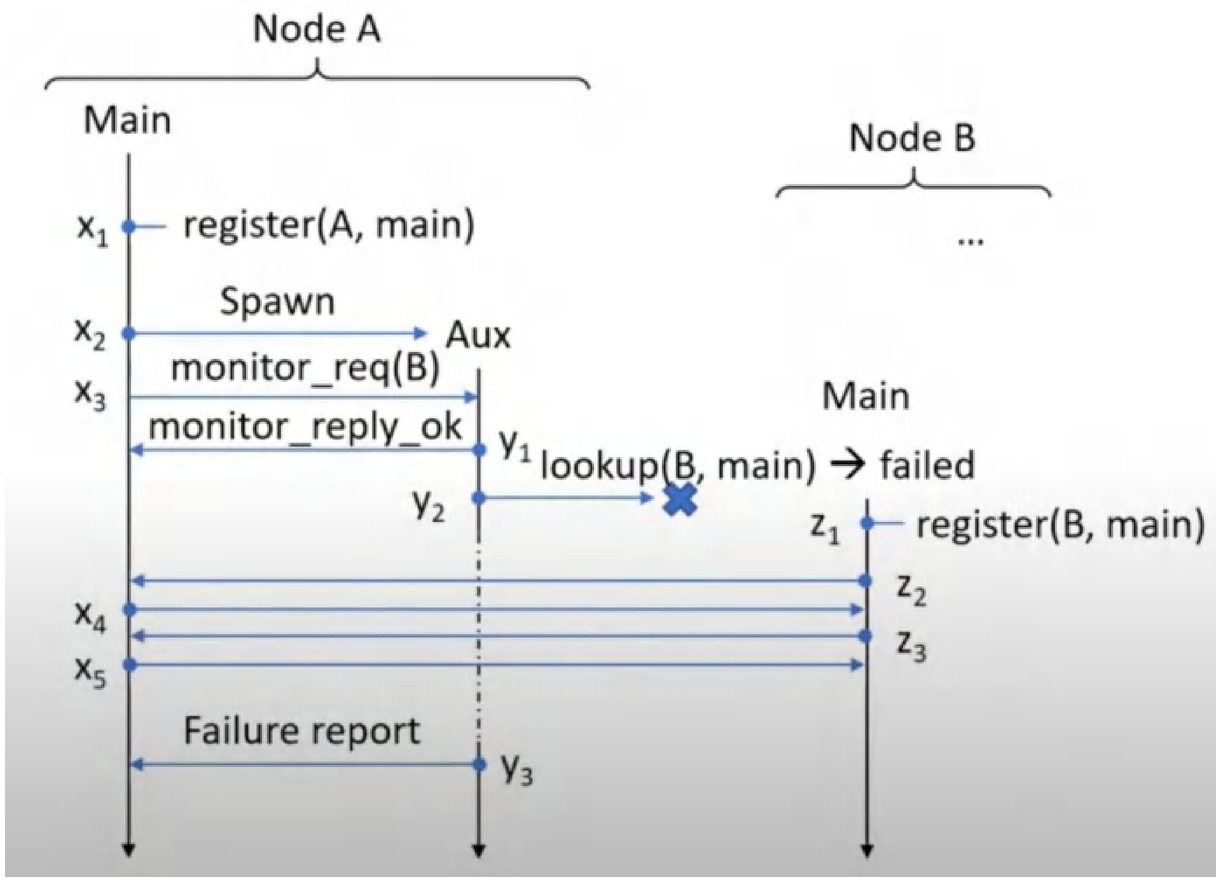

Test start
Node A
Node B
Setting up distributed scenario
Init
Spawn
Init
Spawn
Main
Spawn
Spawn
Aux
Alive? → no
Main
Handshaking
Failure report
Membership: {A}
Membership: {A, B}
Validating system state
Client request
Test error: deadlock!

Partial order sampling (POS)

- Extremely simple algorithm:
  1. Assign operations with random priorities
  2. Always schedule the available operation with the highest

Conflict analysis

- Insight: it suffices to interleave *conflicting operations* to reach any testing result.
  - No need to interleave non-conflicting ones!

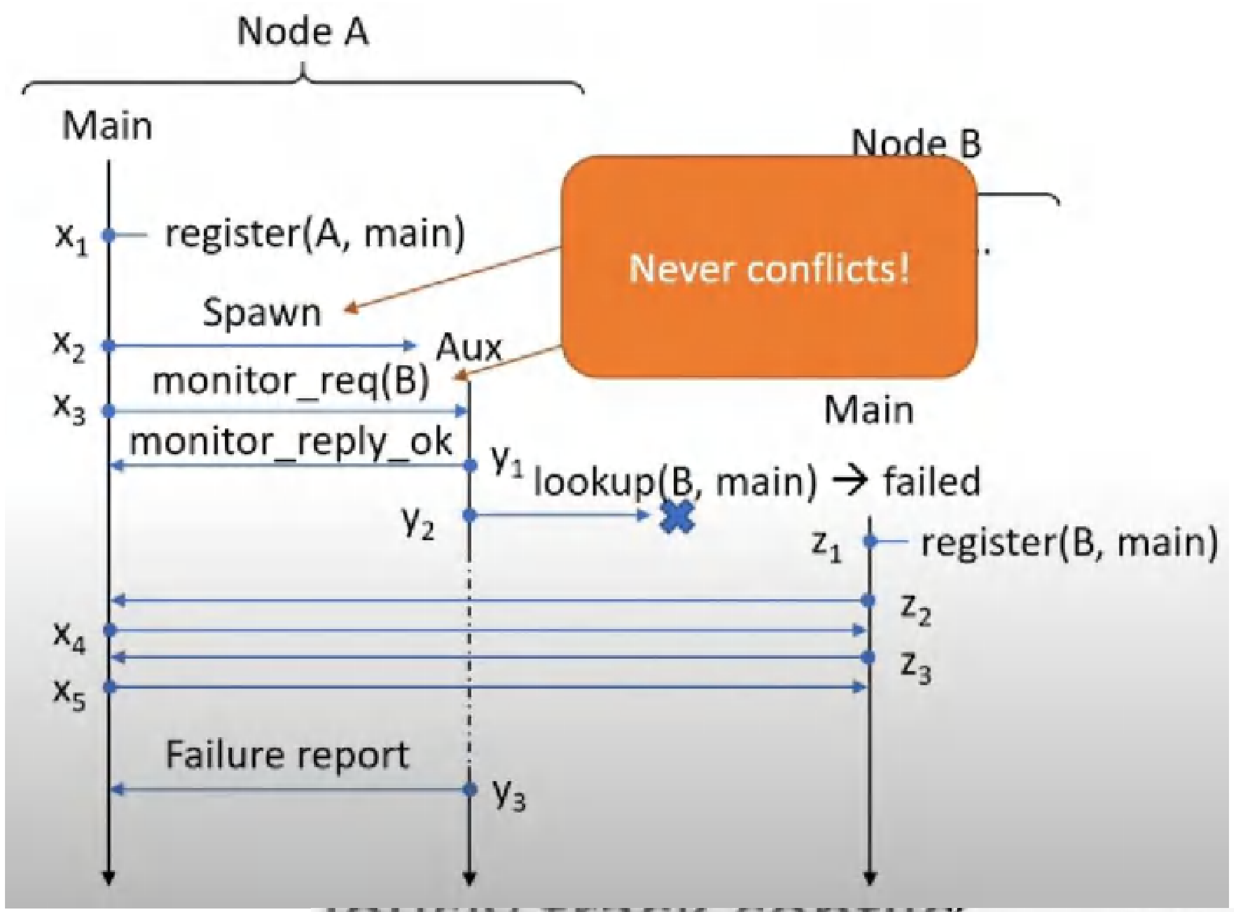


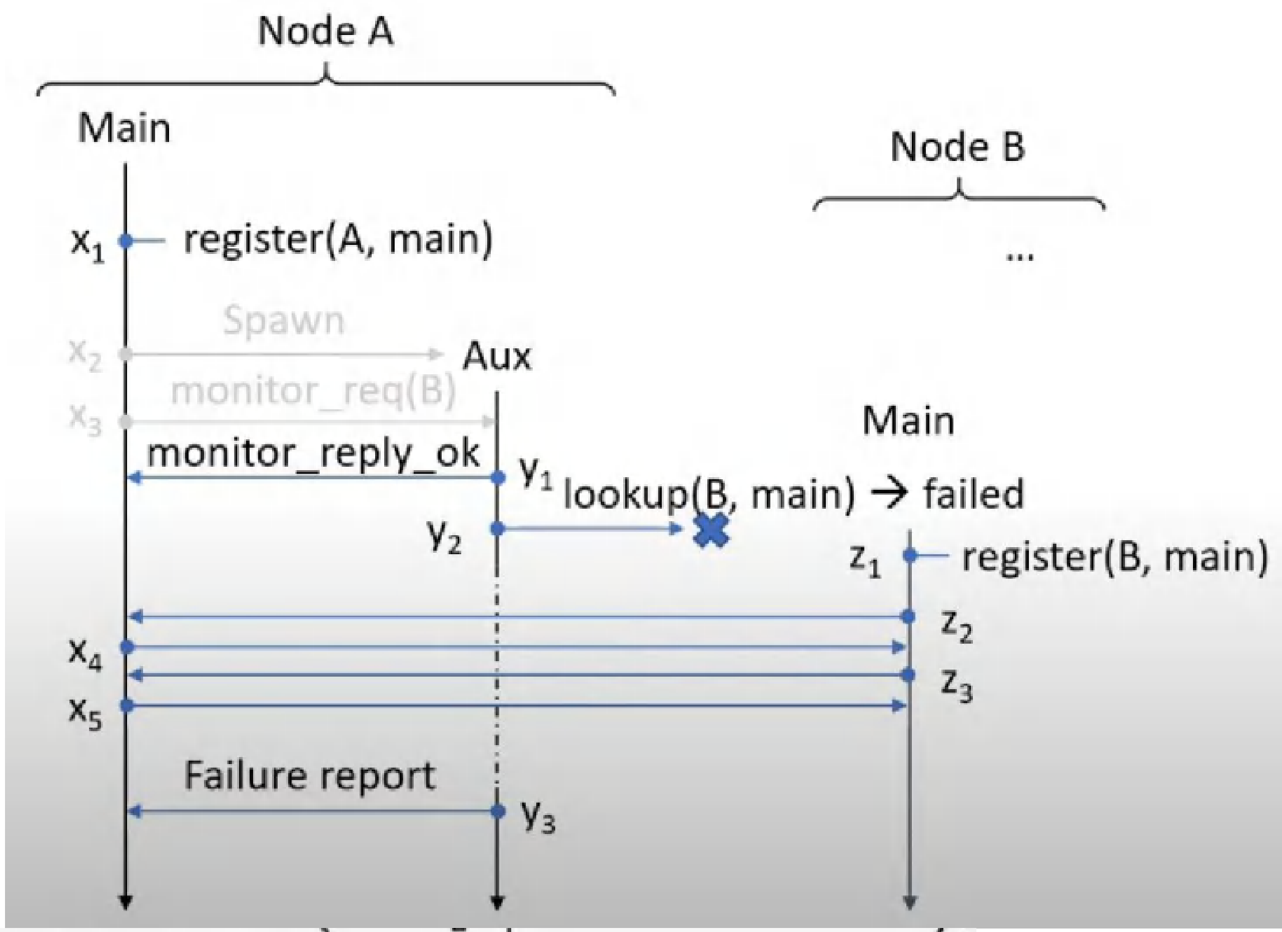
Probabilistic guarantees

$z_1 < \{x_1,x_2,x_3,y_1,y_2\}$

$y_3 < \{x_1,x_2,x_3,x_4,x_5,y_1,y_2,z_1,z_2,z_3\}$

$\Pr = 1/6 \times 1/11 = 1/66$





Probabilistic guarantees

$z_1 < \{x_1, x_2, x_3, y_1, y_2\}$	⇒	$z_1 < \{x_1, y_1, y_2\}$
$y_3 < \{x_1, x_2, x_3, x_4, x_5, y_1, y_2, z_1, z_2, z_3\}$		$y_3 < \{x_1, x_4, x_5, y_1, y_2, z_1, z_2, z_3\}$
Pr = 1/6 x 1/11 = 1/66		Pr = 1/4 x 1/9 = 1/36

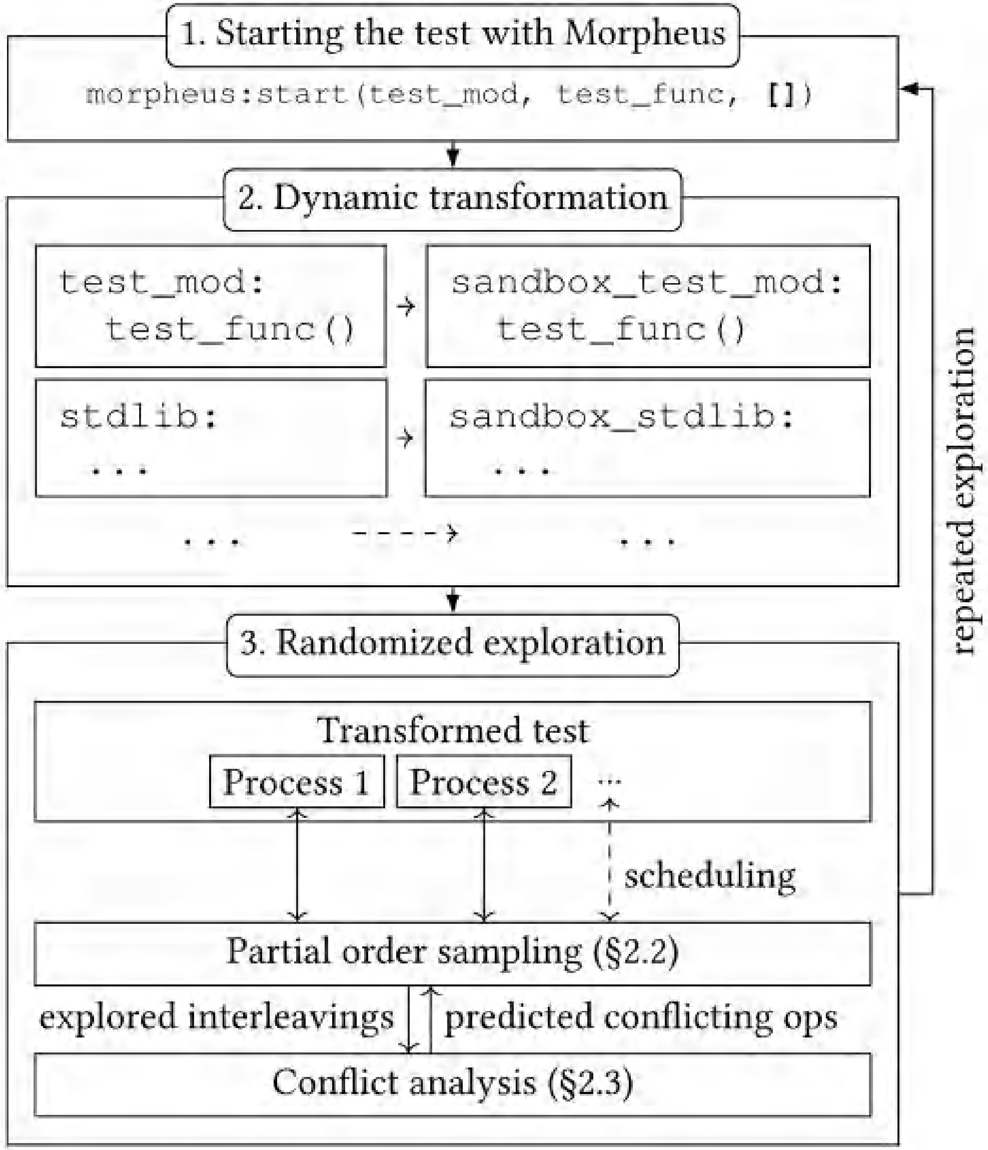
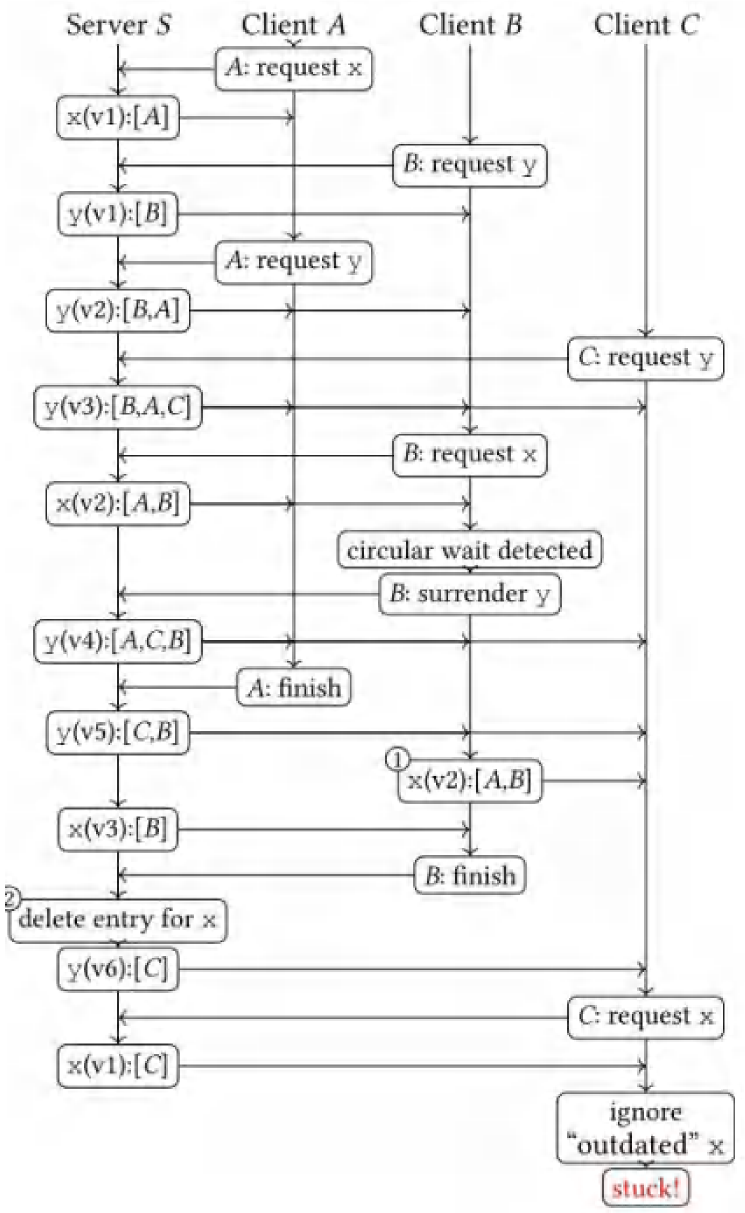
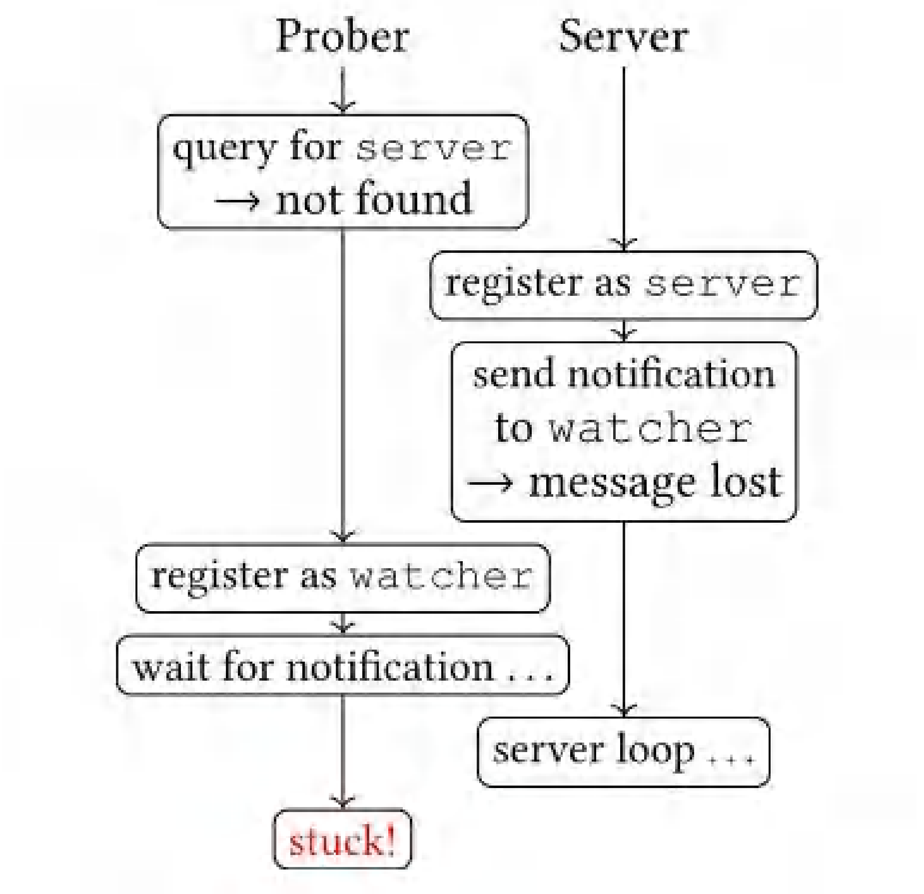


Figure 2. The workflow of Morpheus.

Name	Description	KLOC	Errors
locks	Lock manager	4.1	2
gproc	Process registry	7.3	3
gen_leader	Leader election	1.7	
mnesia	DBMS	27.3	2
rabbitmq	Message broker	60.7	4
ra	Replicated log	8.6	
Total		109.7	11



根本原因：协议设计缺陷。为了性能，协议允许客户端传播其锁条目给其他（可能未请求该锁的）客户端。同时，服务器会在锁空闲时**从本地状态移除该锁的条目以节省空间（重置版本号）**。



复合操作不是原子的，被中间插入的操作打断，导致了状态不一致。

Comparison with Systematic Testing

Case	Systematic	Random walk	POS
crce-2	81	4.29	4.39
crce-3	119	1351.35	17.43

- **实验结果**：
  1. **crce (Chain Replication Protocol)**：
     - `crce-2`：POS (4.39 trials/error) 和 Random Walk (4.29) 性能接近，都远优于系统性测试（81 trials）。
     - `crce-3`：POS (17.43 trials/error) 显著优于 Random Walk (1351.35) 和系统性测试（119）。
  2. **C6023 (Cassandra Bug)**：在10万次试验中，系统性测试和Random Walk**一次都没发现**错误，而**POS发现了20次。**
  3. **locks-1**：只有这个真实错误能在Concuerror上运行。结果：系统性测试（0/100000）、Random Walk (1/100000)、**POS (18987/100000, 约5.27 trials/error)。**

Morpheus Error-detection Performance

Case	RW	RW+	PCT	PCT+	POS	POS+	POS*	POS*+
locks-1	0.0042	0.0312	0.0895	0.1164	0.1521	0.2087	0.1562	0.2239
locks-2	0.0210	0.0117	0.0022	0.0071	0.0073	0.0124	0.0103	0.0140
gproc-1	0	0.0001	0.0015	0.0018	0.0031	0.0106	0.0008	0.0023
gproc-2	0	0.0190	0.0496	0.0781	0.0605	0.1170	0.0416	0.0648
gproc-3	0	0.0002	0.0431	0.0385	0.0156	0.0450	0.0027	0.0094
mnesia-1	0	0.0001	0.0219	0.0223	0.0141	0.0168	0.0091	0.0124
mnesia-2	0	0.0008	0.0075	0.0078	0.0117	0.0253	0.0104	0.0254
ms-1	0	0.0061	0.3000	0.2833	0.1489	0.2420	0.1505	0.2478
ra-1	0	0	0.0025	0.0036	0.0070	0.0120	0.0062	0.0126
ra-2	0	0	0.0010	0.0011	0.0053	0.0042	0.0063	0.0044
ra-3	0	0.0001	0.0003	0.0002	0.0032	0.0031	0.0032	0.0033
Mean	N/A	N/A	0.0089	0.0106	0.0151	0.0249	0.0109	0.0180
Ratio to POS+	N/A	N/A	35.62%	42.66%	60.67%	100.00%	43.74%	72.01%
CA Improvement		N/A		19.77%		64.82%		64.65%

最佳组合：POS+ (基础POS + 冲突分析) 是表现最好的组合。

总体优势：POS+ 的整体错误检测率是 RW 的 **280.77%**，是 PCT 的 **234.90%**。优势极其明显。

POS*的意外表现：高级版POS (POS*) 的表现反而比基础版 (POS) 更差（平均低28%），这验证了作者在§2.2中的判断：在消息传递模型中，复杂的依赖跟踪会因误报而性能下降。

RW的致命缺陷：Random Walk (RW) 在11个错误中有**2个根本检测不到**（hit-ratio为0），这印证了引言中关于其概率保证极弱的论述。

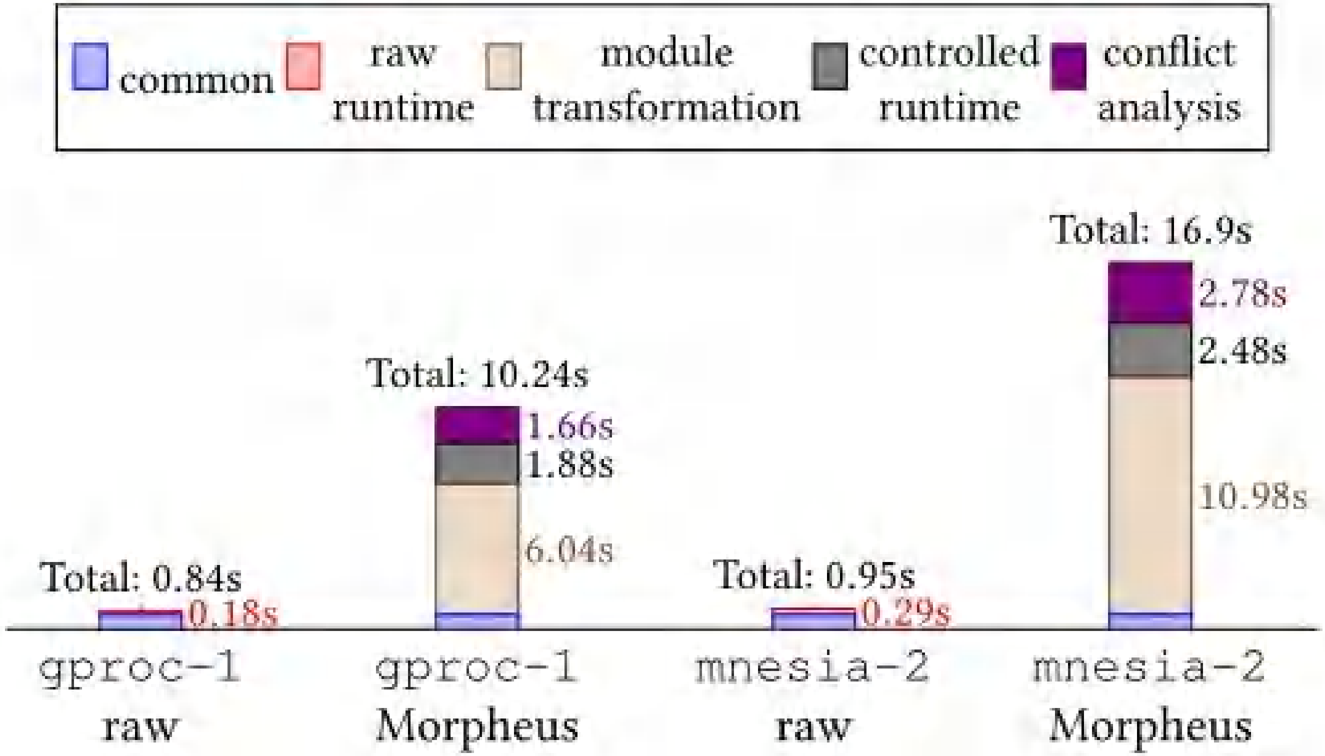
Effect of Conflict Analysis

Case	Operations	Conflicts	Hit-ratio
gproc-1	6593.39	325.59	0.0031
Scheme	FNs	FPs	
PC	0.09	1192.16	0.0064 (206%)
{P,PC}	0.54	526.27	0.0106 (342%)
mnesia-2	10018.80	628.25	0.0117
PC	0.18	4612.02	0.0174 (149%)
{P,PC}	1.13	1442.74	0.0253 (216%)

（P，PC）签名将**误报（False Positives）**的非冲突操作数量大幅降低（从1192->526, 4612->1442），使其与真实冲突数处于同一量级。

更好的精度直接转化为**更高的性能**：使用（P，PC）签名后，冲突分析对 gproc-1 和 mnesia-2 的检测性能分别提升了 **342%** 和 **216%**（与无CA的POS相比）。

The Real-time Performance of Morpheus



1. **模块重写 (Module Transformation)**：占比最大。这是"一次性"开销，可以通过缓存优化。
2. **受控运行时 (Controlled Runtime)**：执行插桩后的代码带来的开销。
3. **冲突分析 (Conflict Analysis)**：每个试验后分析轨迹的开销。

Morpheus引入了**近两个数量级的开销**。这在需要完全控制交错的一致性测试中是**常见且可接受的**。