Understanding Prediction-Based Partial Redundant Threading for Low-Overhead, High-Coverage Fault Tolerance

Vimal Reddy

Sailashri Parthasarathy†

Eric Rotenberg

Dept. of Electrical & Computer Engineering
North Carolina State University
Raleigh, NC

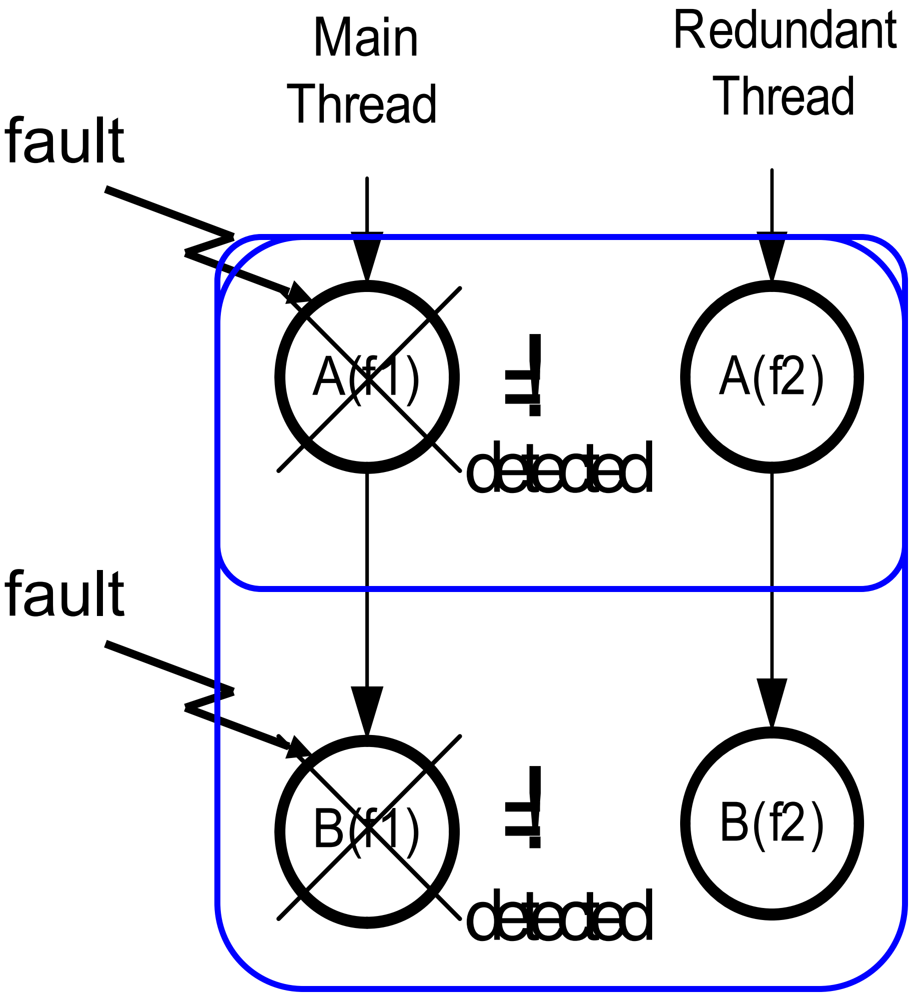
† Architecture Modeling Infrastructure Group
Intel Corporation
Hudson, MA

Transient Fault Tolerance

- Transient faults
  - Temporary hardware faults
  - Worsening with shrinking technology
    - Soft errors
    - Noise

- Prominent solution: Redundant Multithreading
  - Full program duplication
  - Complete fault tolerance
  - High overheads

Full Redundant Execution

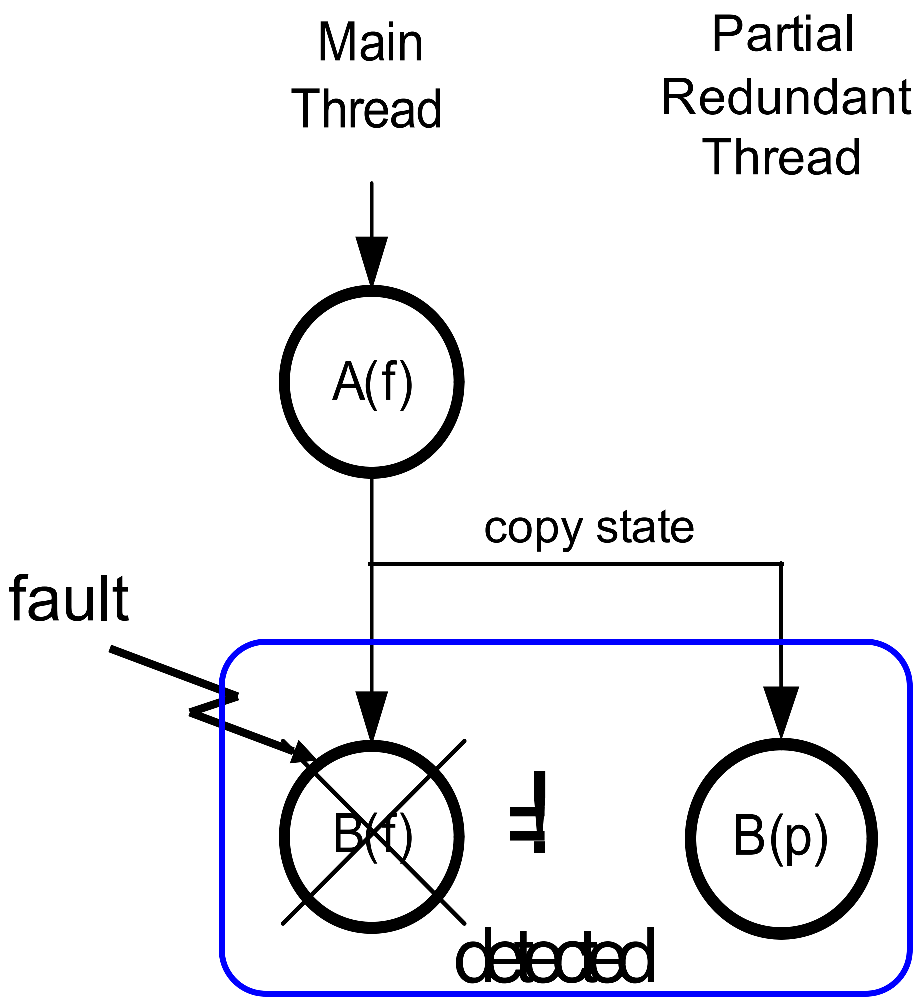


- Full duplication
- 100% fault coverage

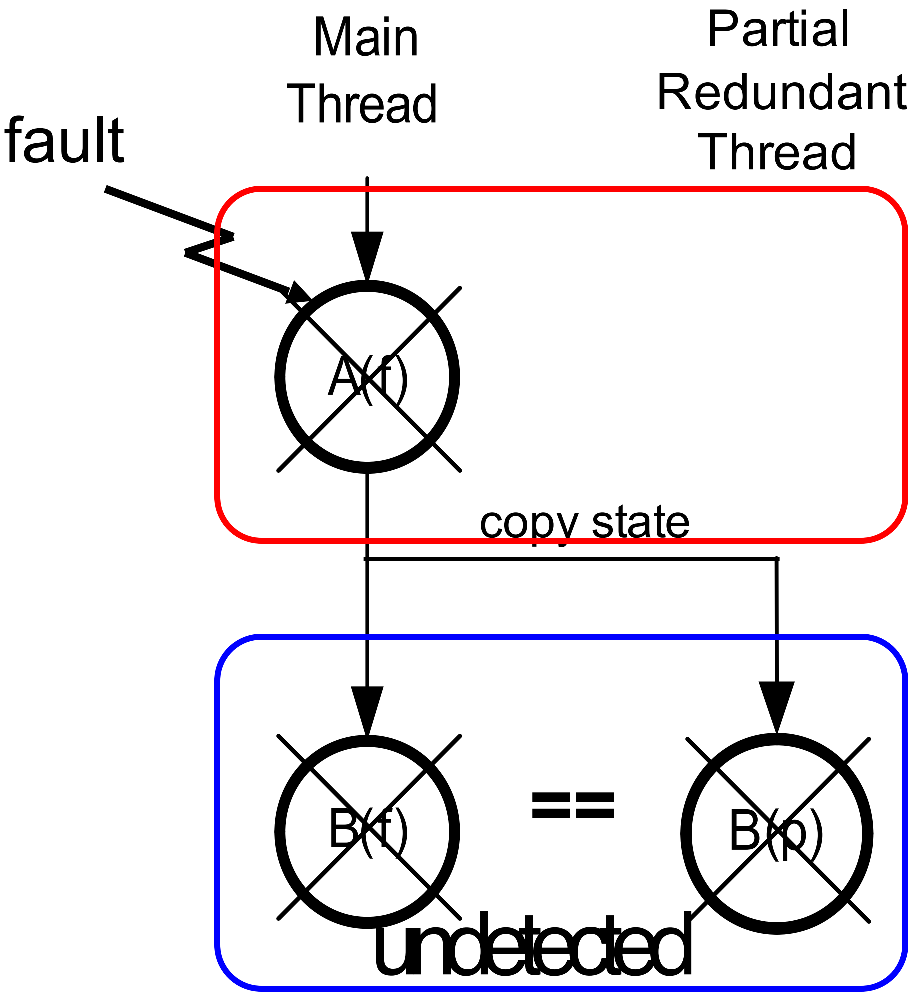
Partial Redundant Threading (PRT)

- Only partially duplicate a program
- Shorter the redundant thread, lesser the overhead
- Approach taken to create partial thread affects fault tolerance

Conventional PRT

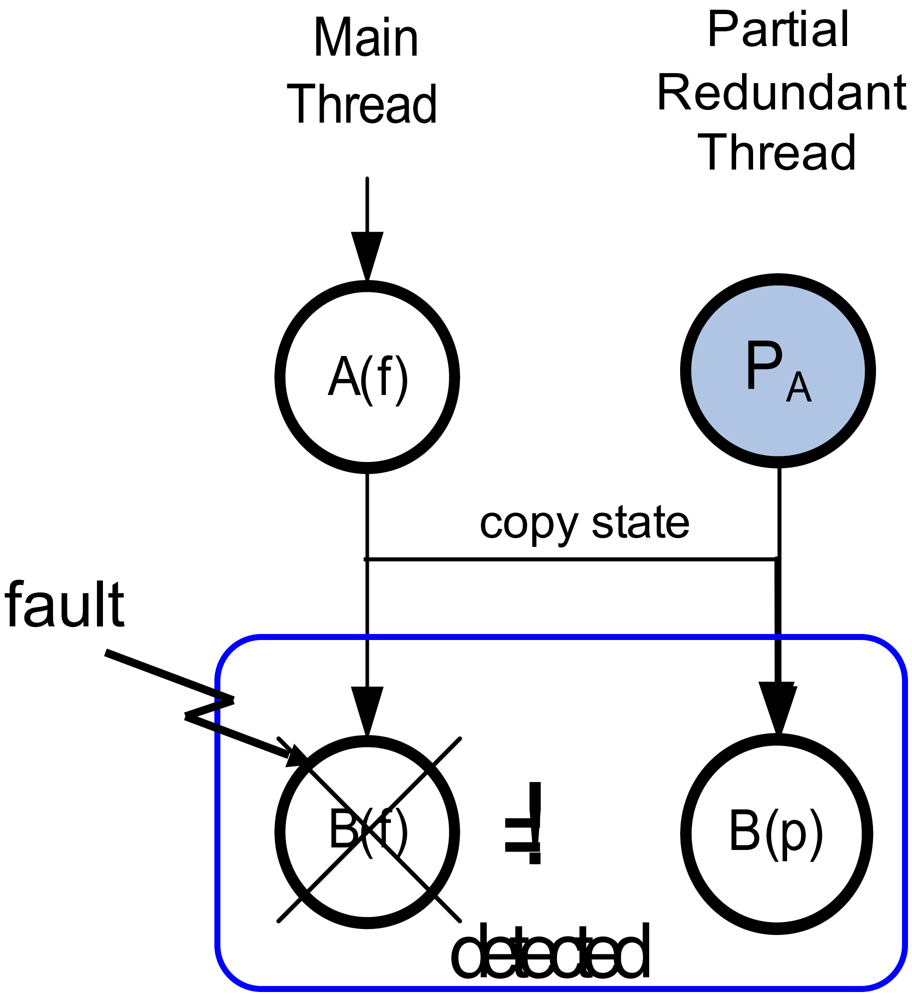


Conventional PRT



- Arbitrary duplication
- Non-duplicated portions los fault coverage

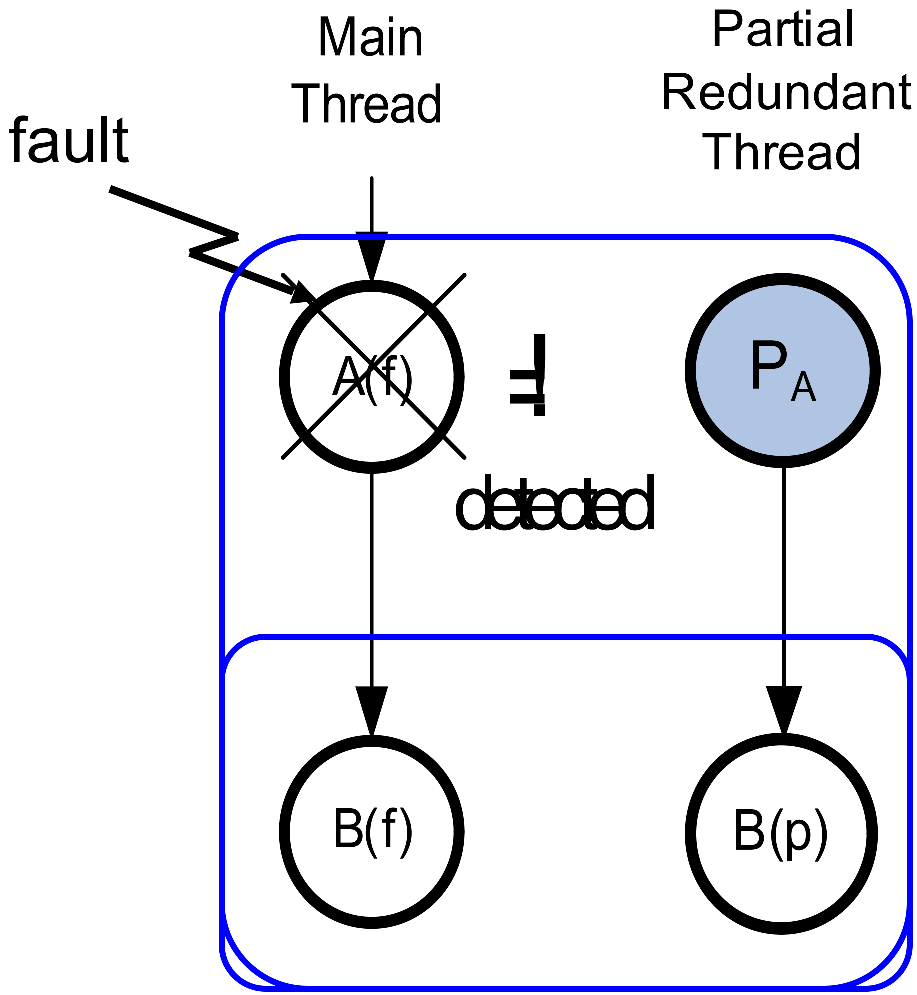
Prediction-based PRT



Confident prediction of A

Freq. Case: Correct prediction (e.g., 99.9% of the time)

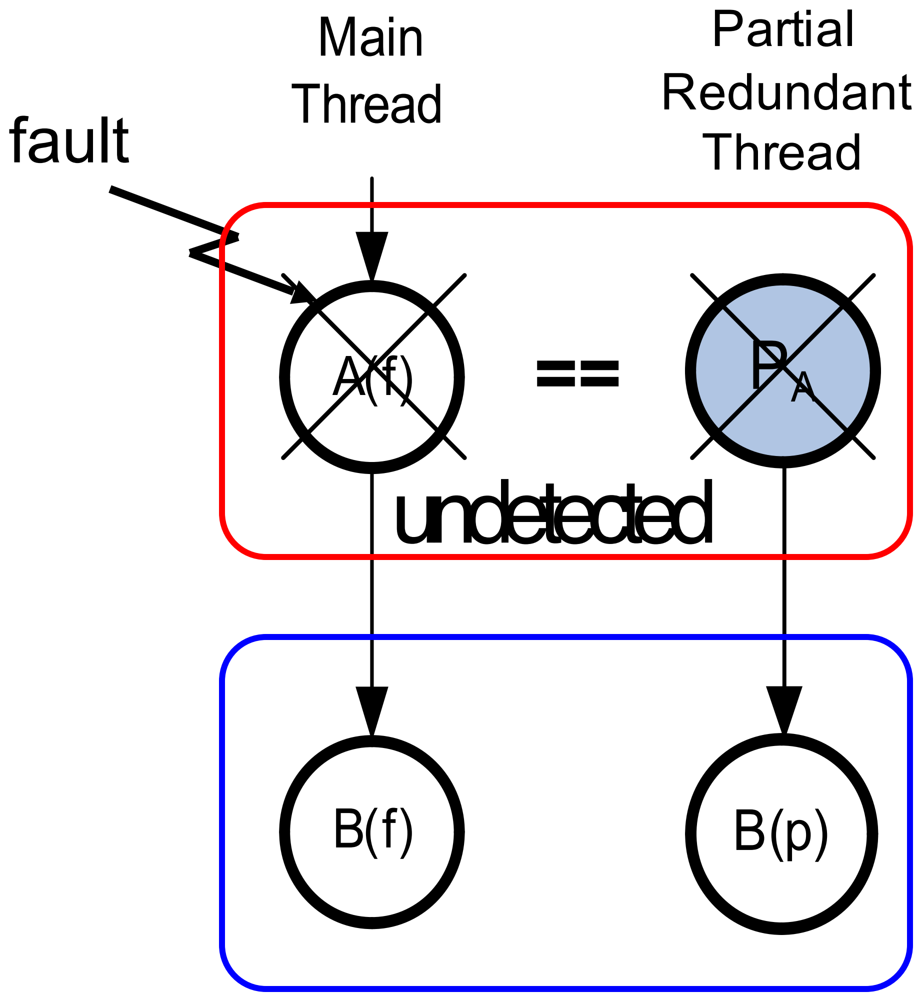
Prediction-based PRT



Confident prediction of A

Freq. Case: Correct prediction (e.g., 99.9% of the time)

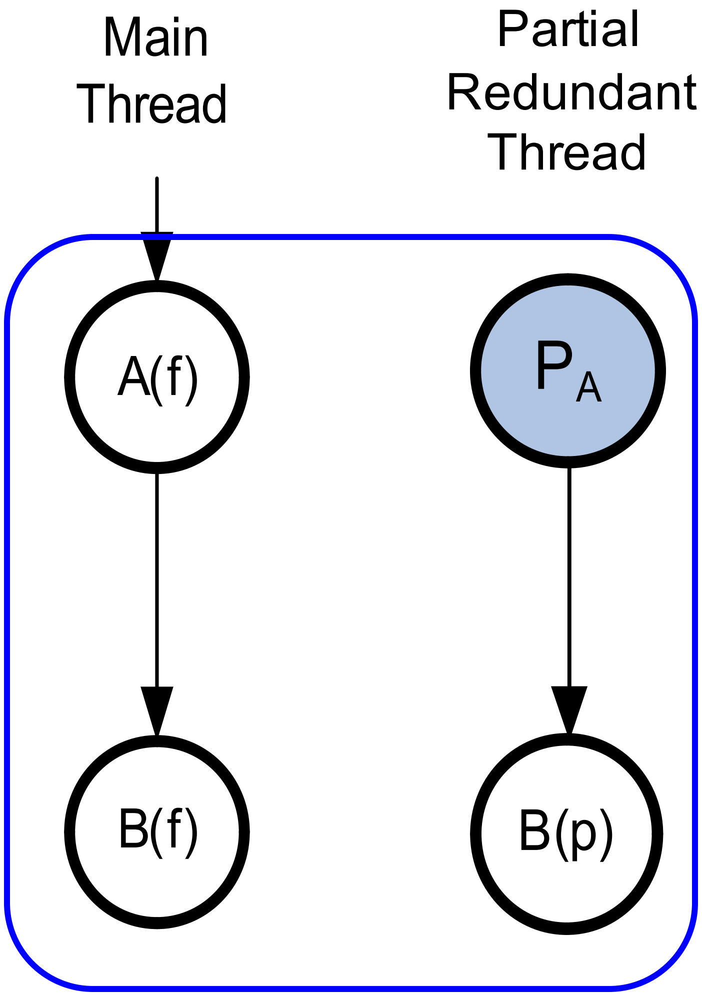
Prediction-based PRT



Confident prediction of A

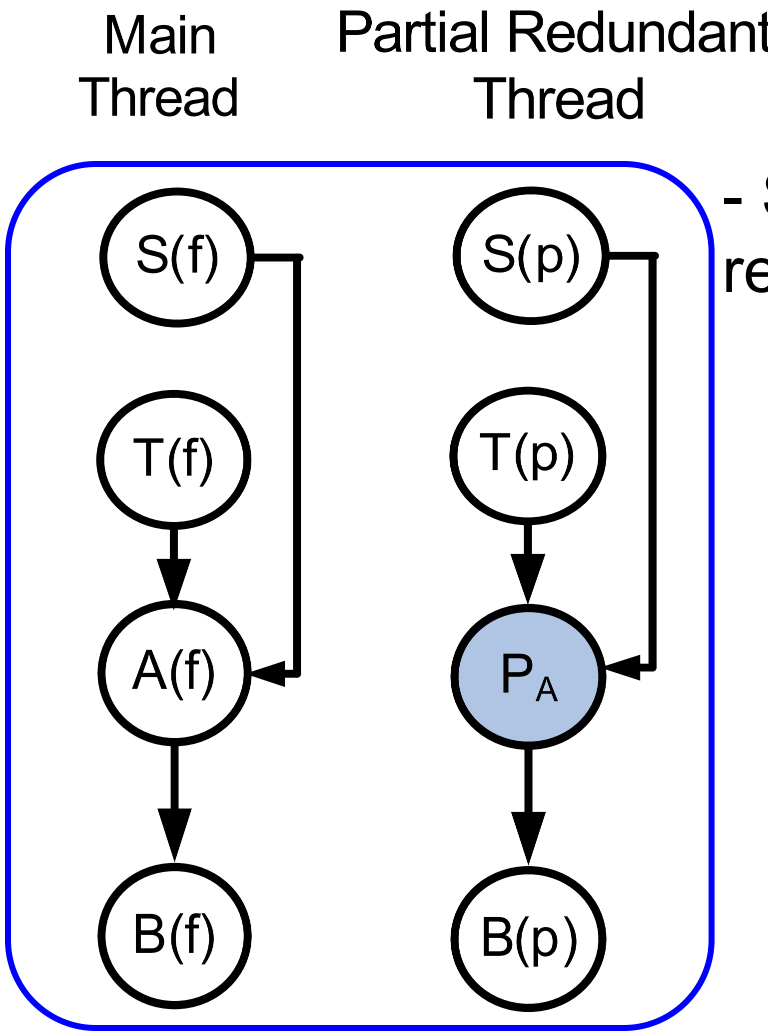
Rare case: Incorrect prediction (e.g., 0.1% of the time)

Prediction-based PRT



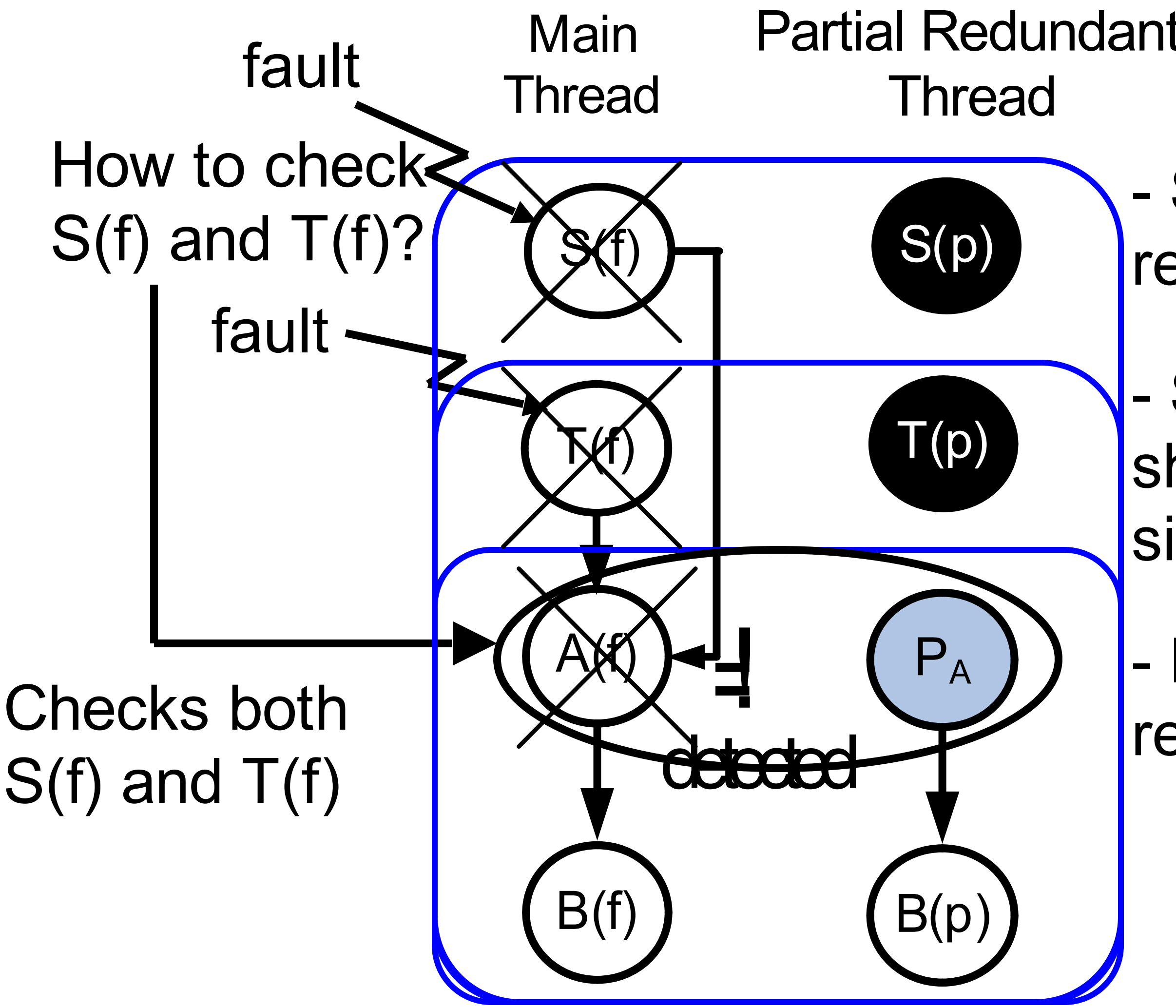
- Confident predictions are good **proxies** for redundant execution
- Predictions break thread inter-dependence
- Near-100% fault coverage

Relaxing checking constraints



- S(p) and T(p) are removable

Relaxing checking constraints



- S(p) and T(p) are removable

- Such removal can shorten partial thread significantly

- But, no predictions to replace them

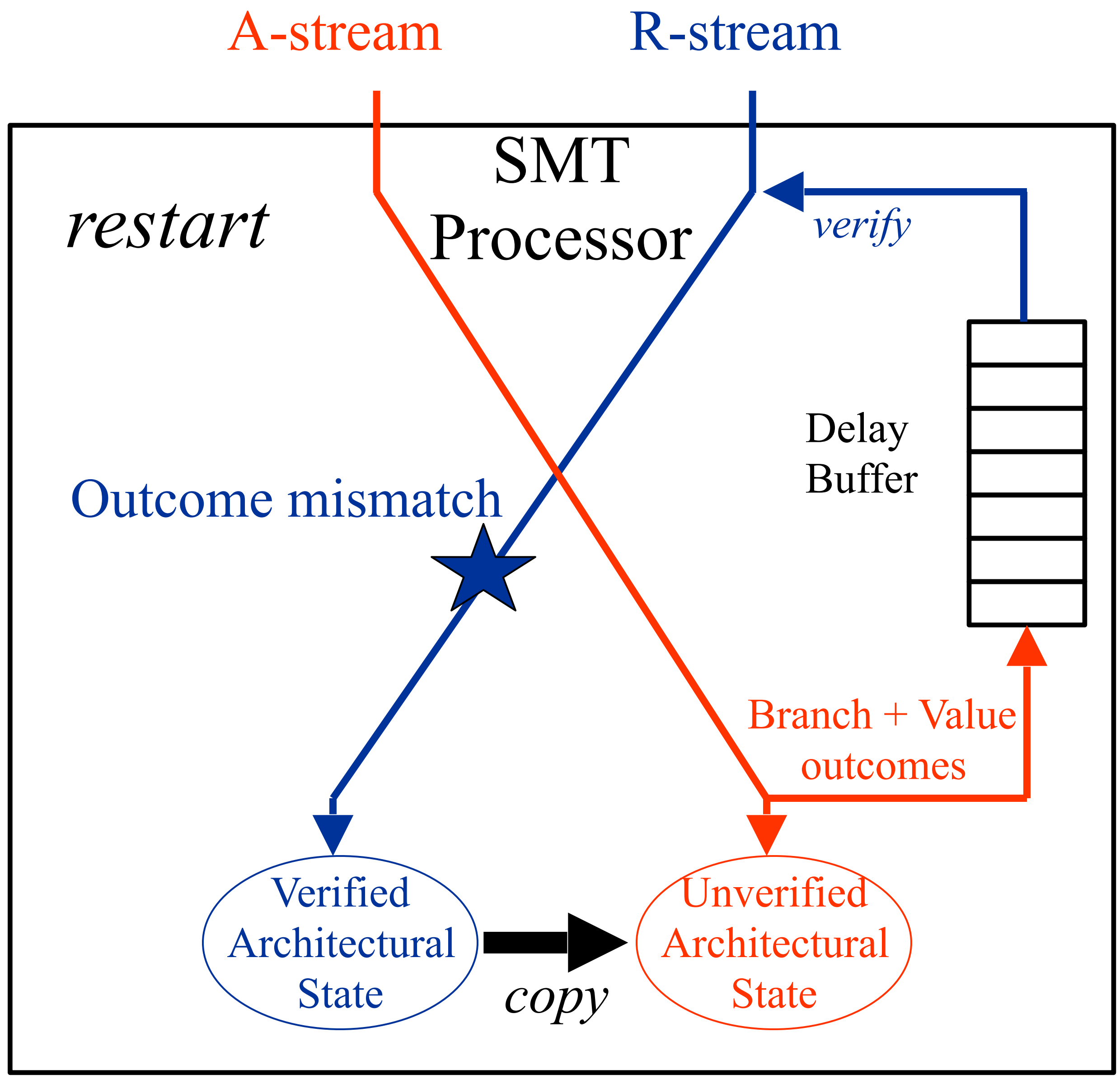
PRT Spectrum

PARTIAL REDUNDANT THREADING (PRT) SPECTRUM

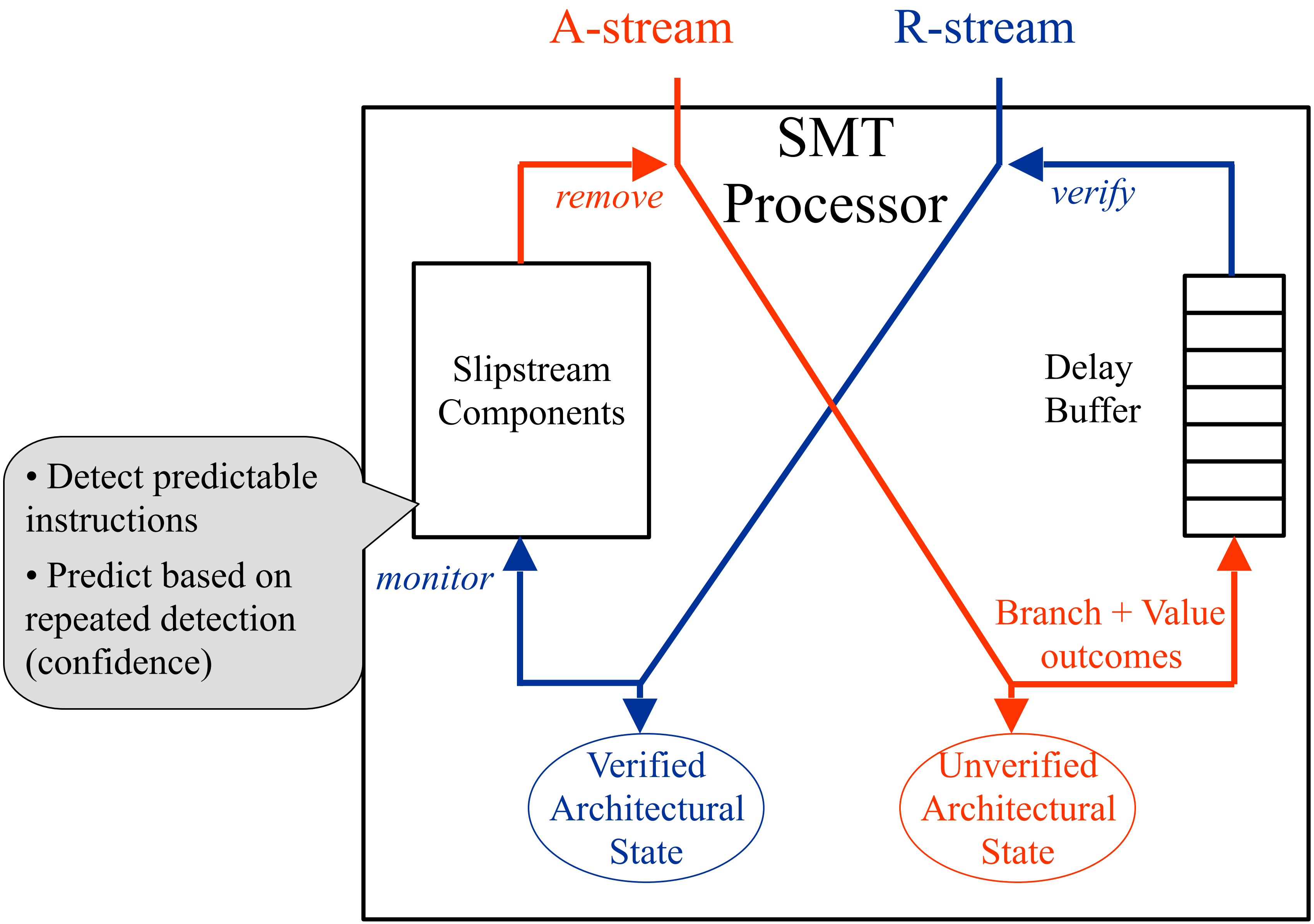
Partial Duplication	Partial Duplication & Confident Predictions	Confident Predictions
EX: Opportunistic	EX: Slipstream	EX: ReStore
M. Gomaa T. N. Vijaykumar	Z. Purser K. Sundermoorthy E. Rotenberg	N. Wang S. J. Patel
ISCA 2005	MICRO 2000 ASPLOS 2000	DSN 2004

- Case study: PRT on Slipstream

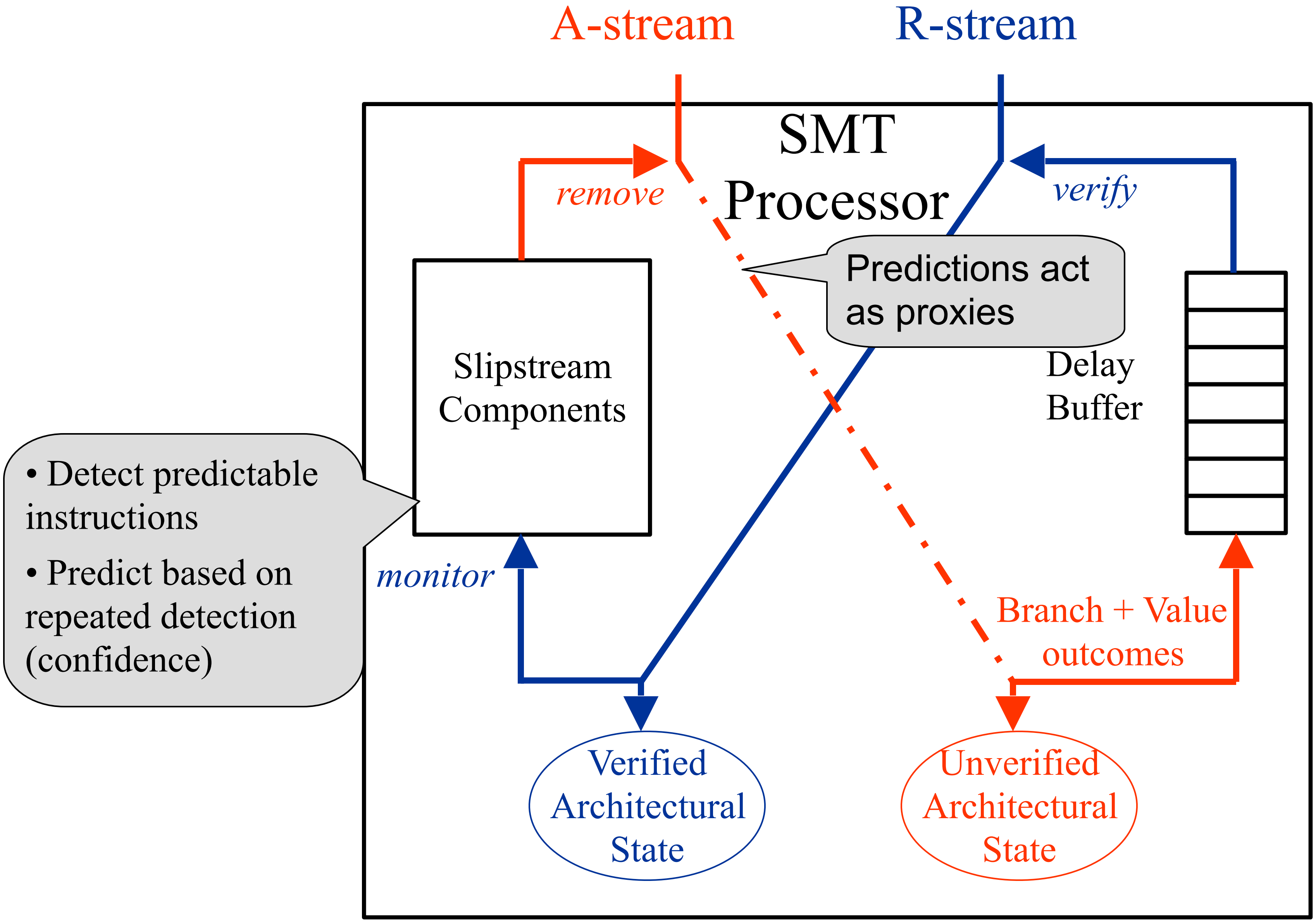
Slipstream Overview



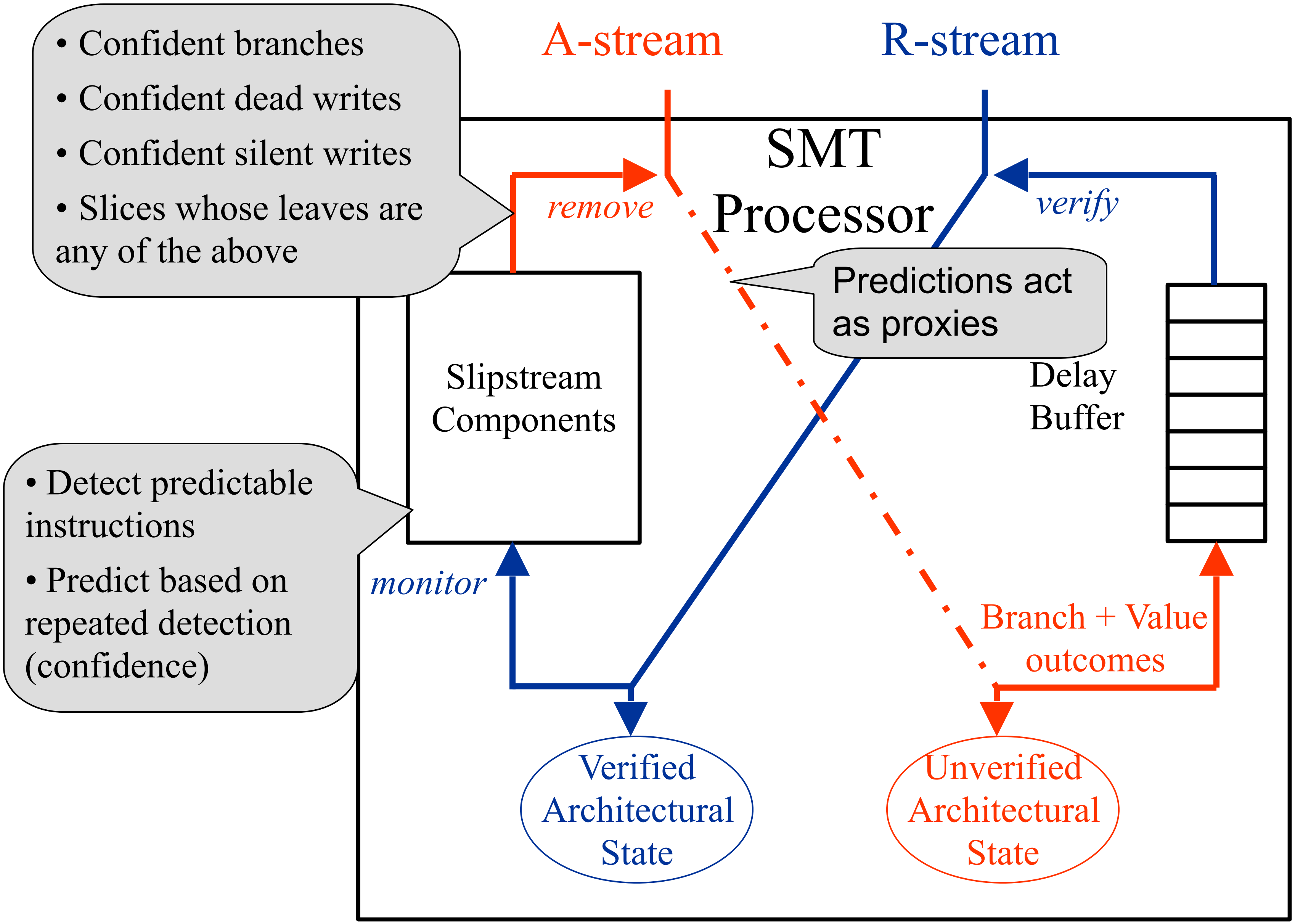
Slipstream Overview



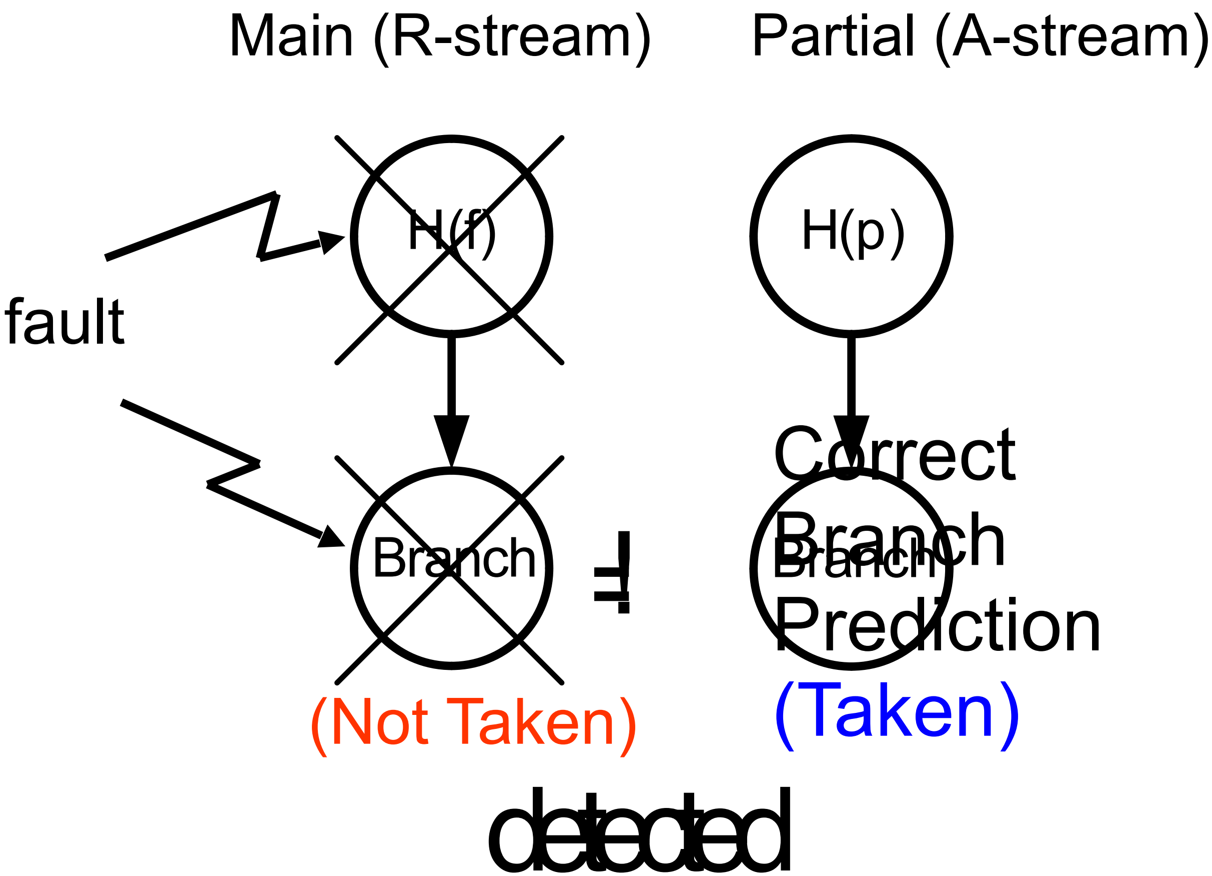
Slipstream Overview



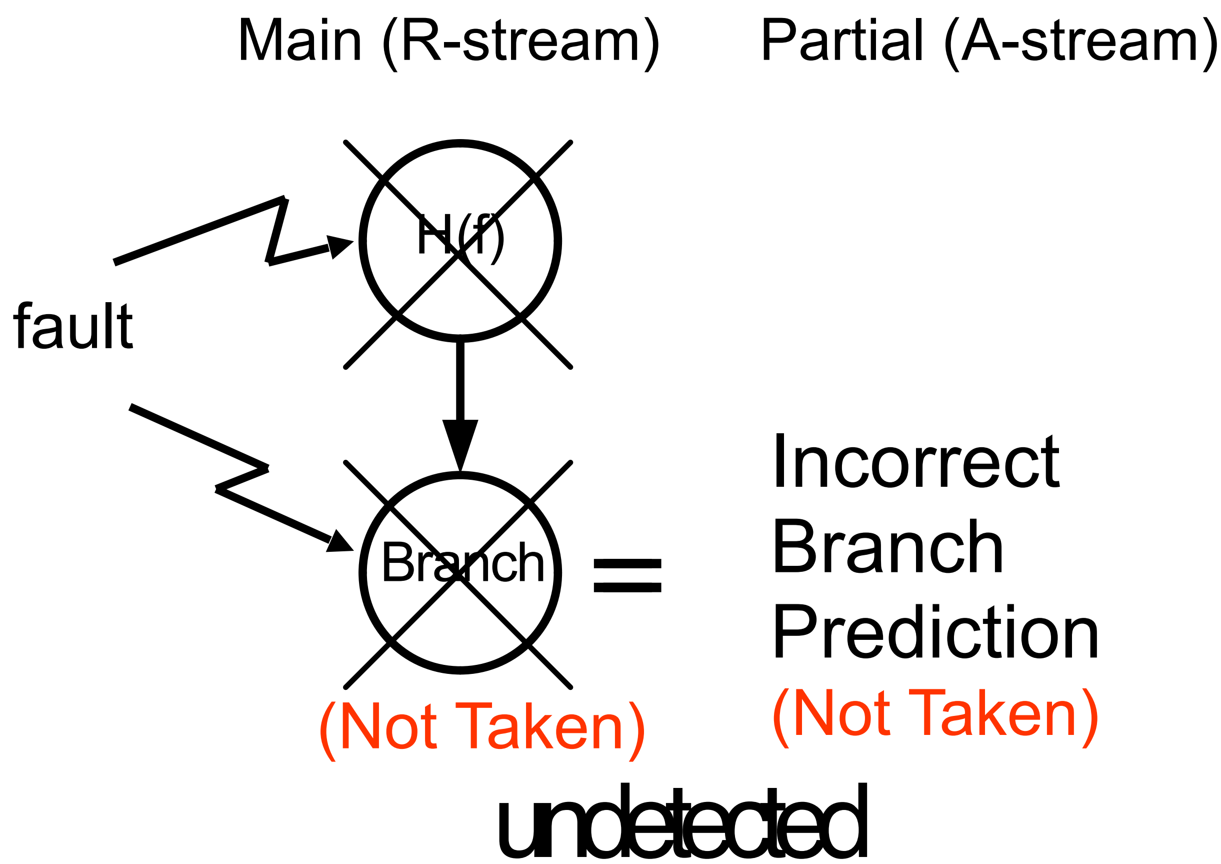
Slipstream Overview



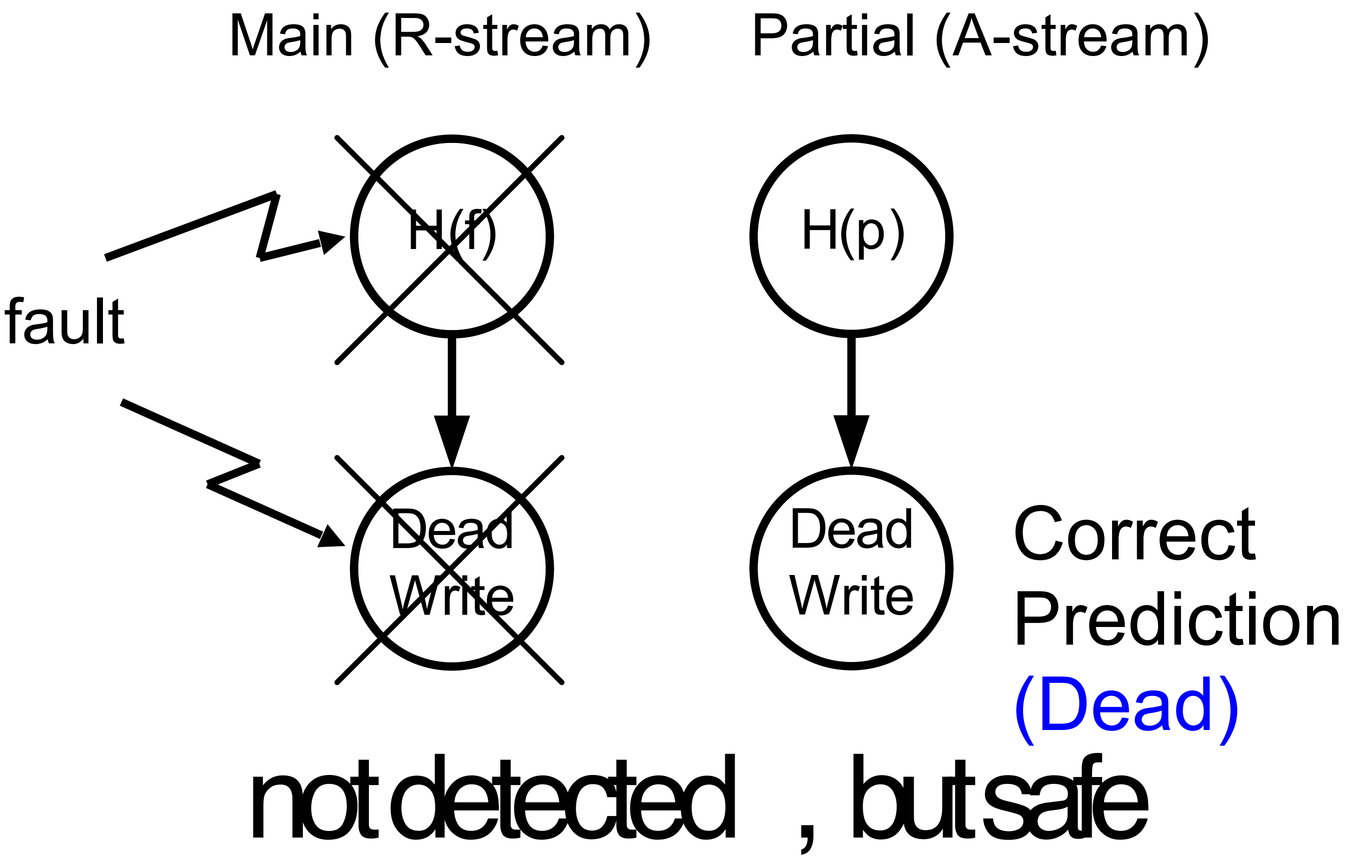
Confident Branch



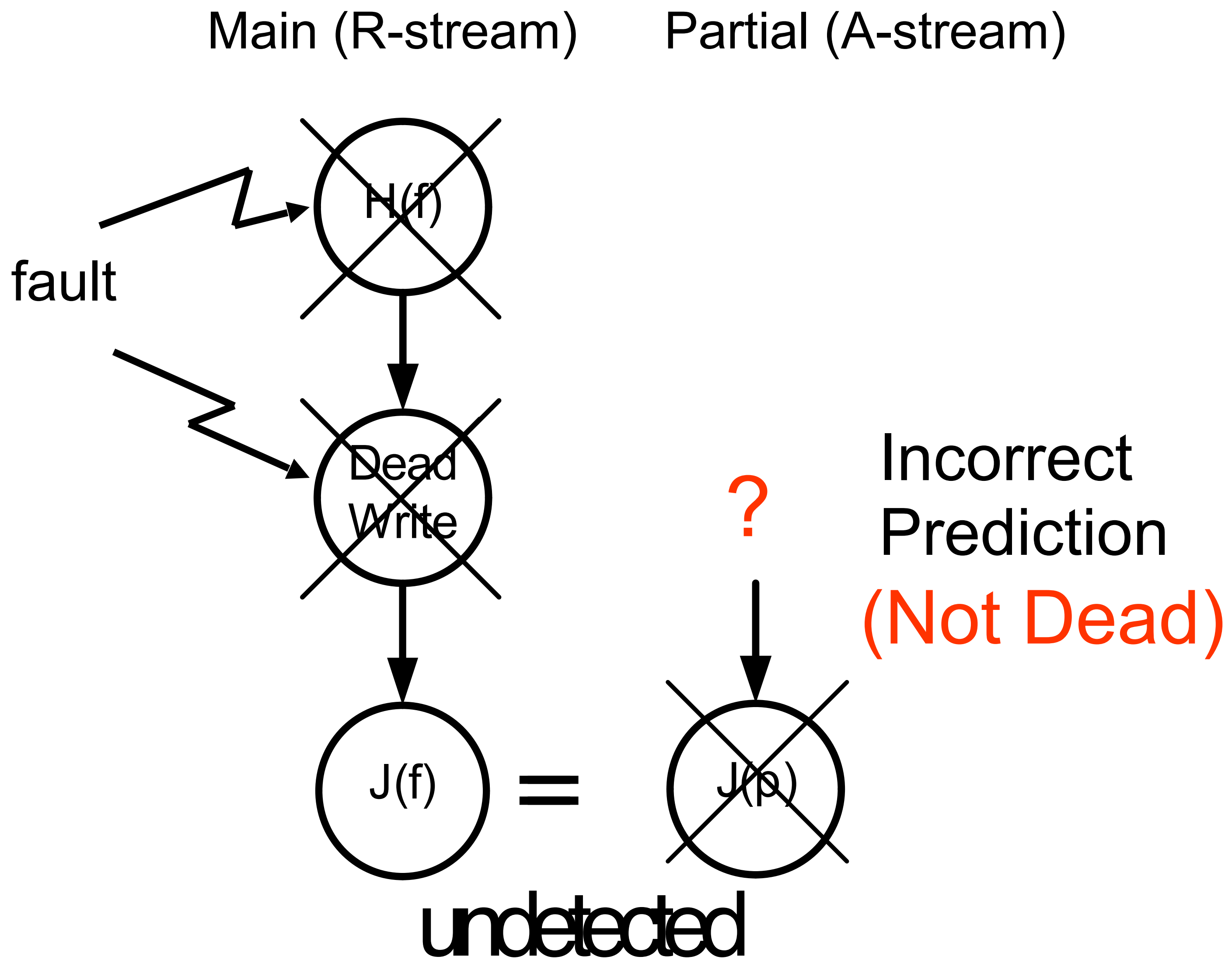
Confident Branch



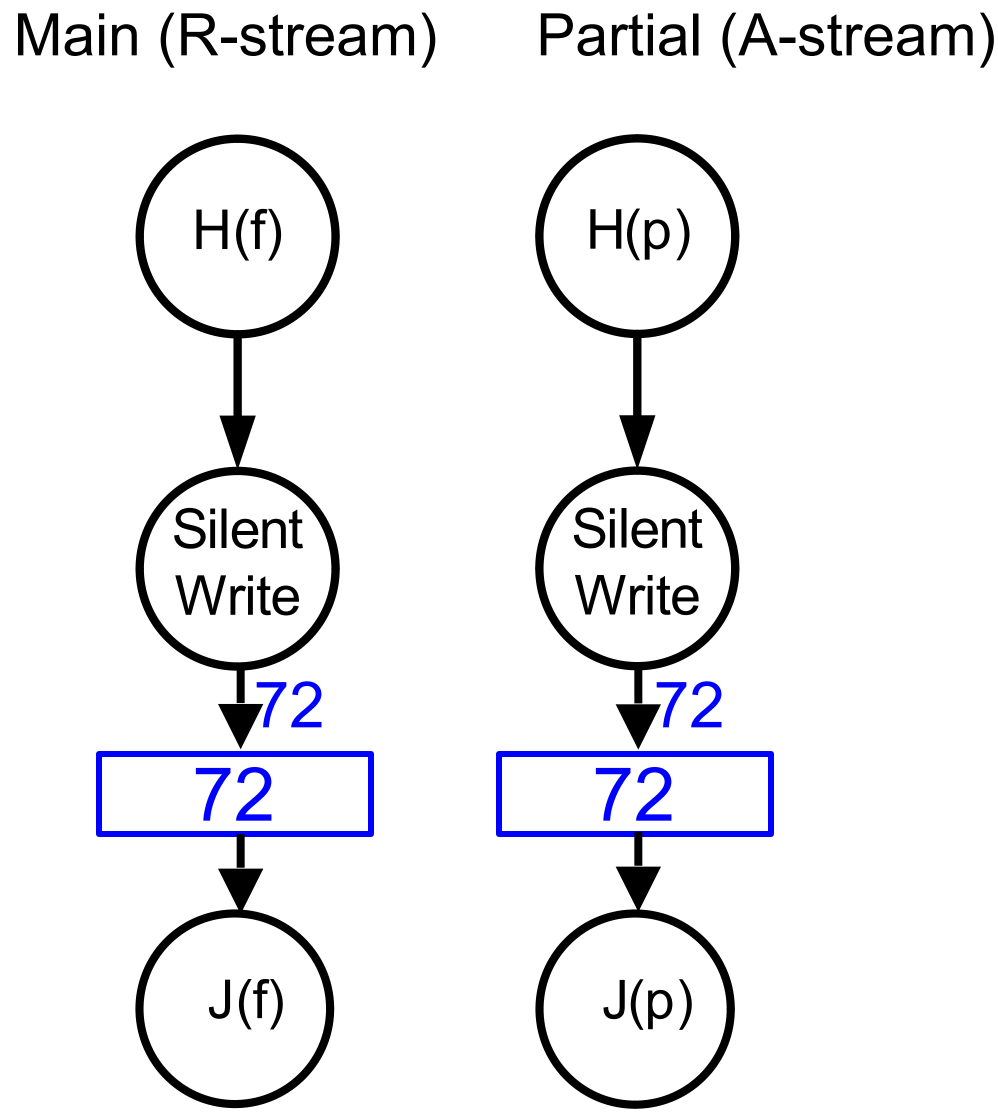
Confident Dead Write



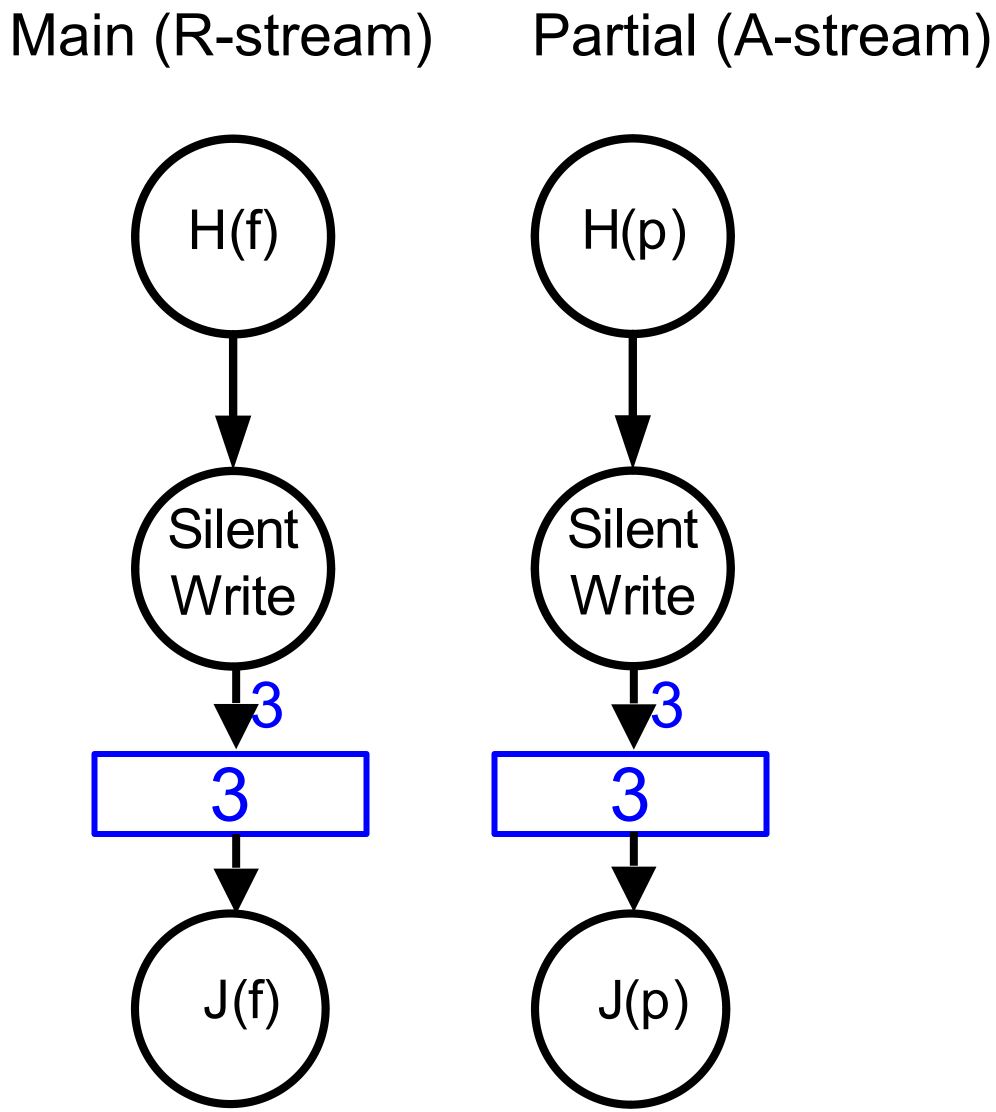
Confident Dead Write



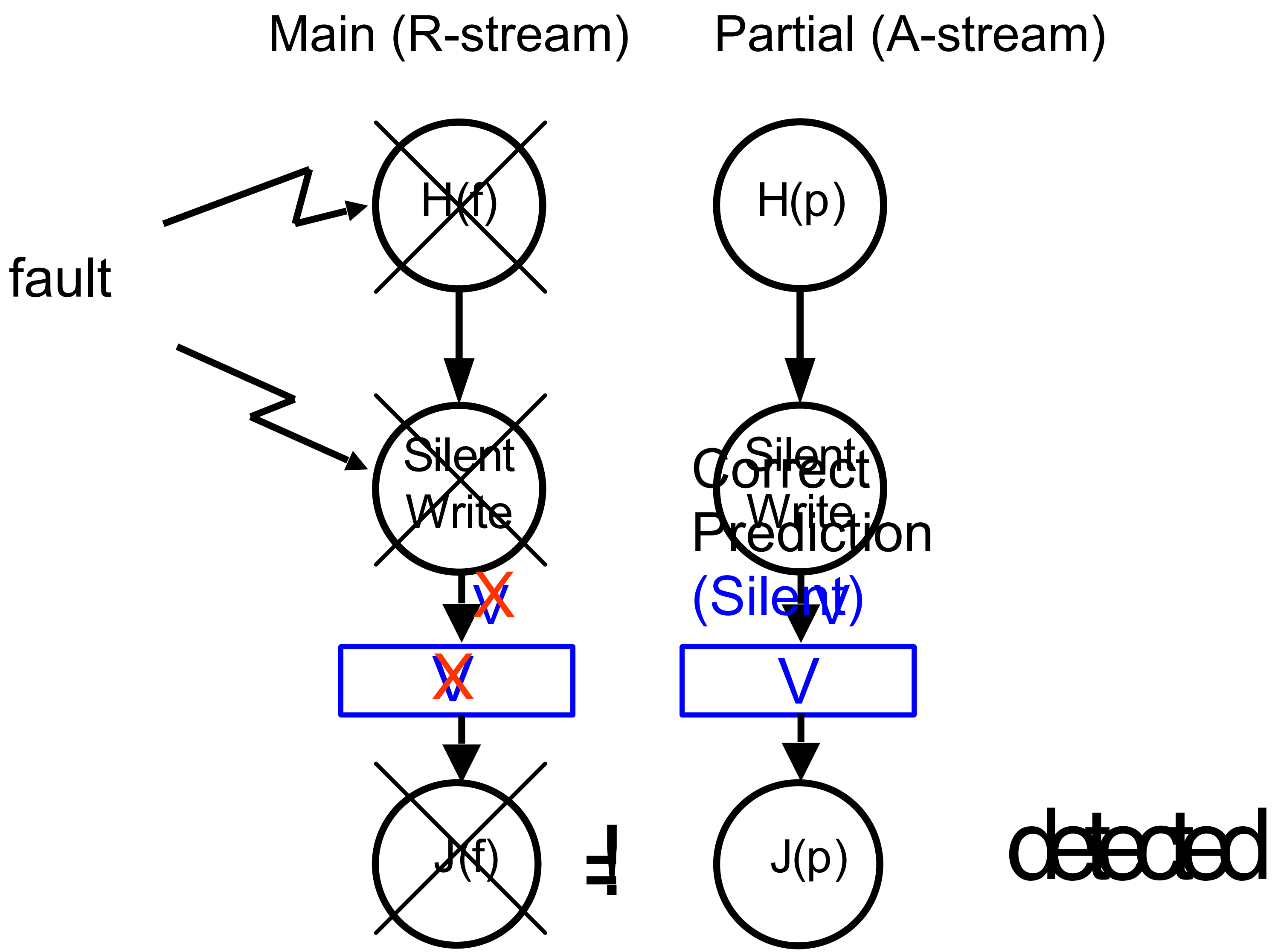
Confident Silent Write/Store



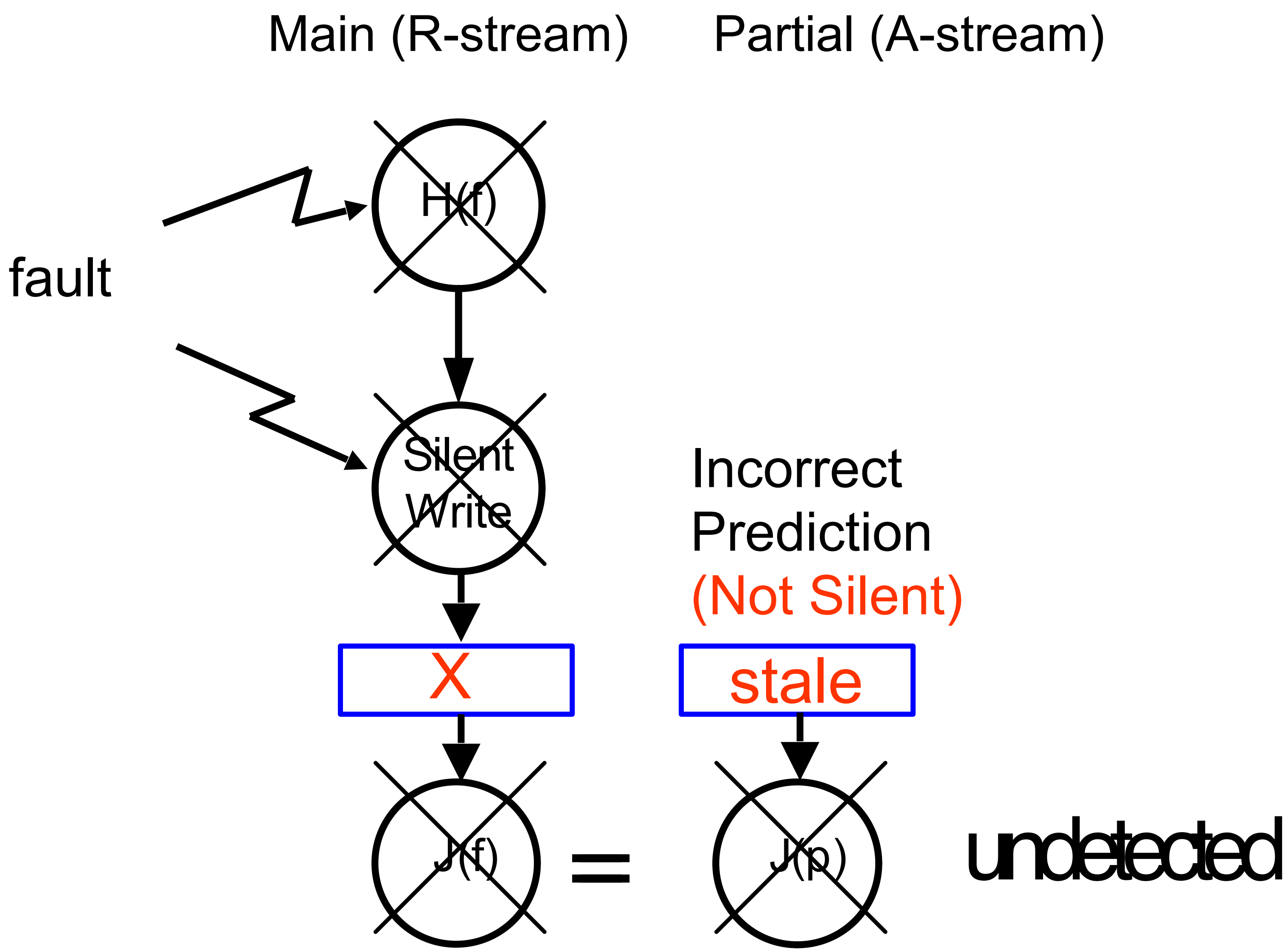
Confident Silent Write/Store



Confident Silent Write/Store



Confident Silent Write/Store



Slipstream Fault Detection Coverage

- Prior research: Only duplicated instr. covered
- We showed confident predictions detect faults
  - Additional coverage =
    Correctly predicted confident instr. + Their backward slices
  - Mispredictions vulnerable, but rare in Slipstream
  - Mispredictions + backward slices = only 0.1% instr.
- Hence, non-duplicated instr. well covered

Slipstream has high fault detection coverage (99.9%)

Fault Coverage: Detection vs. Recovery

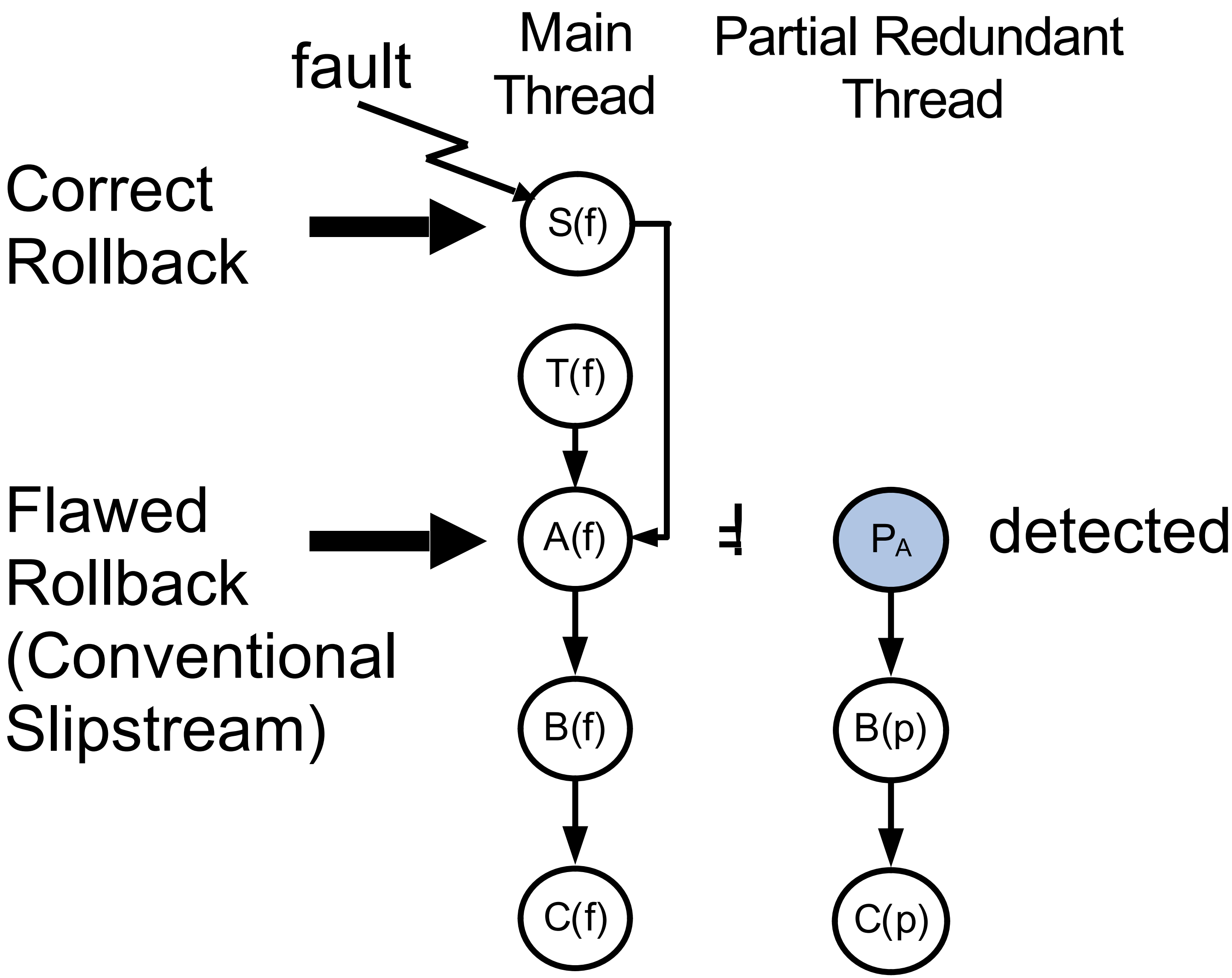
Detection Coverage

- Represents ability to detect faults
- Slipstream: 99.9% of instr.

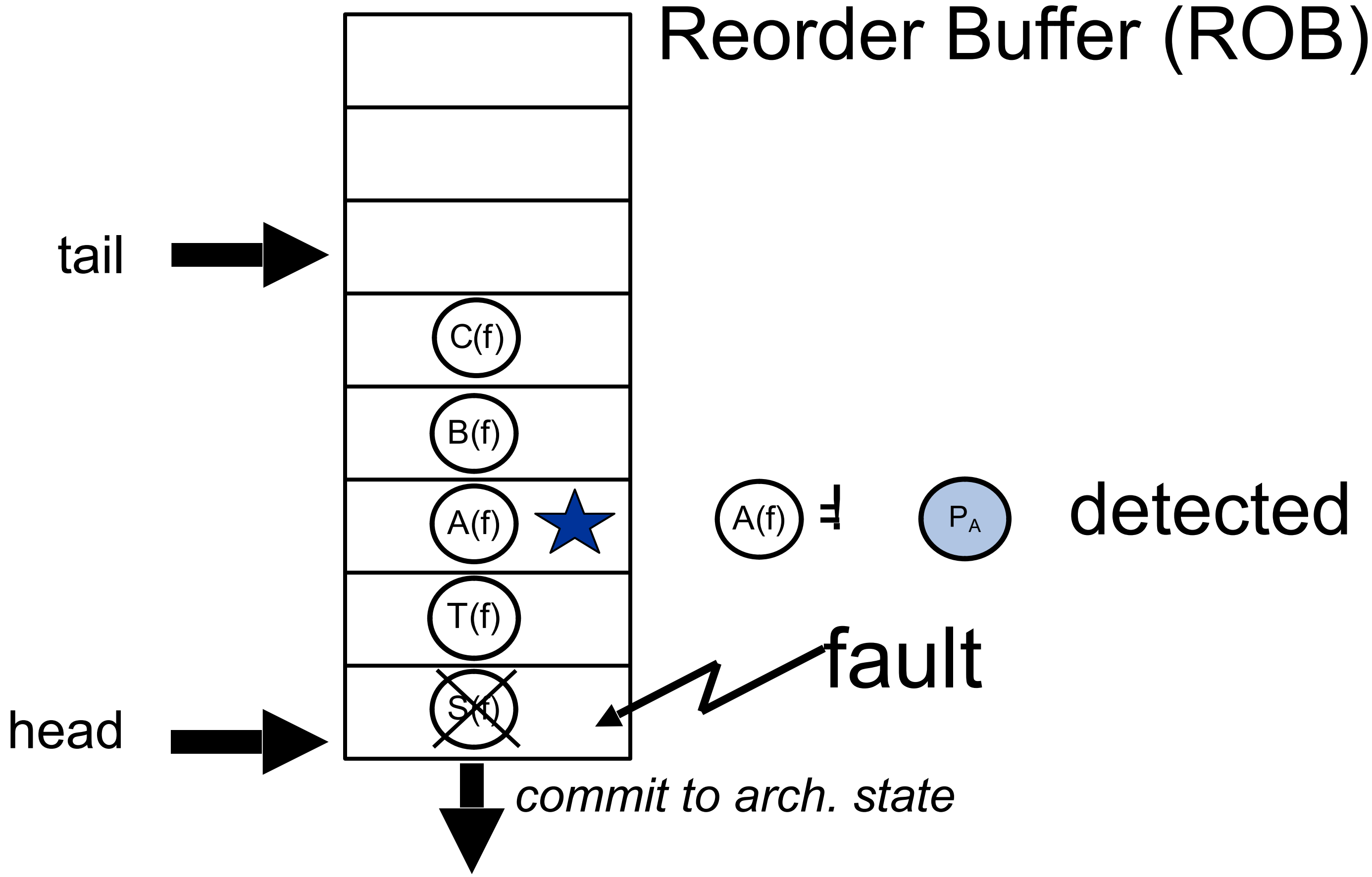
Recovery Coverage

- Ability to rollback to 'golden' state on fault detection
- Slipstream's recovery coverage?

Slipstream Fault Recovery



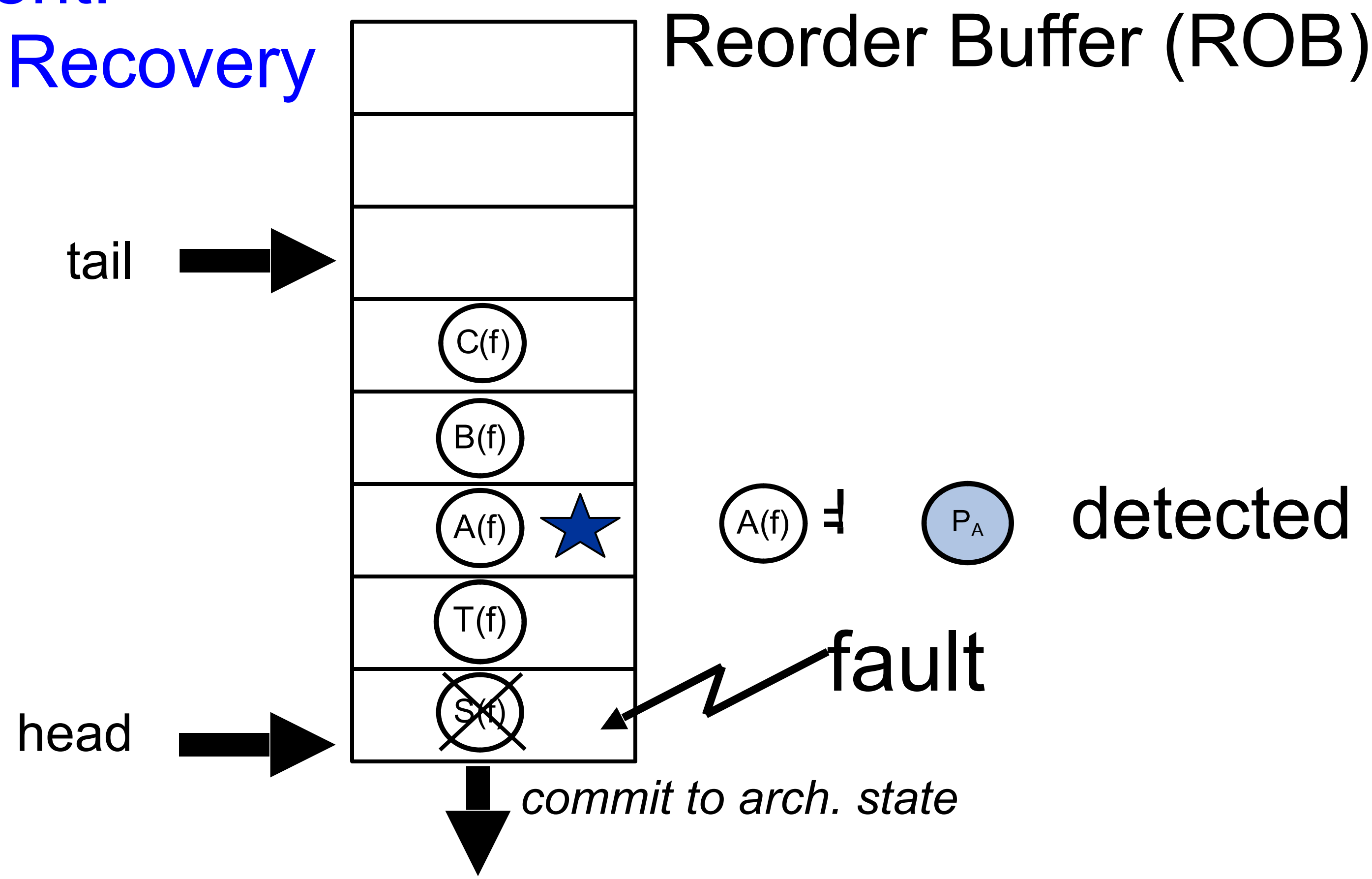
Base Slipstream Recovery



Flush
Restart

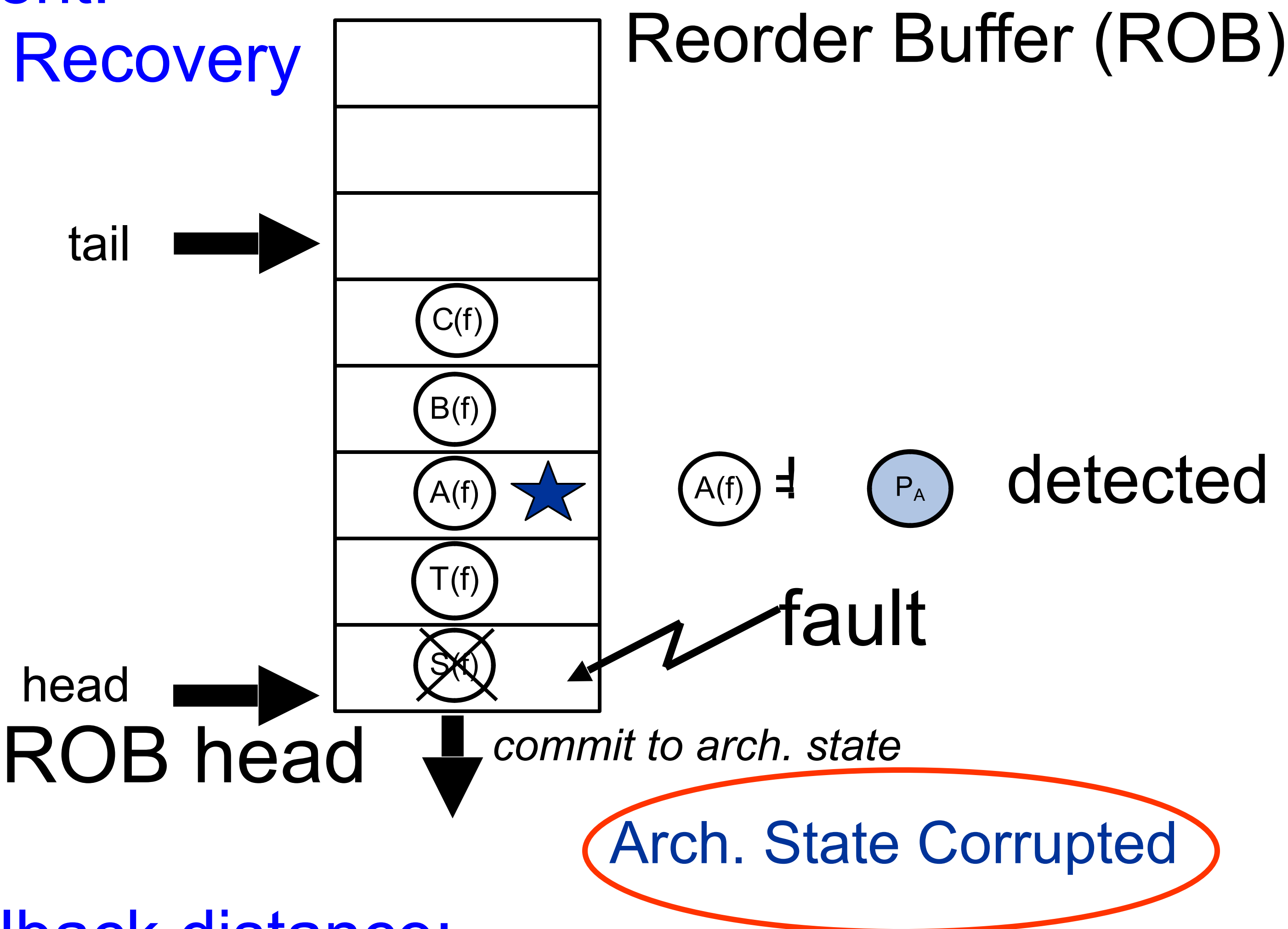
Arch. State Corrupted

Enhancement: ROB-head Recovery



Flush to ROB head
Restart

Enhancement: ROB-head Recovery

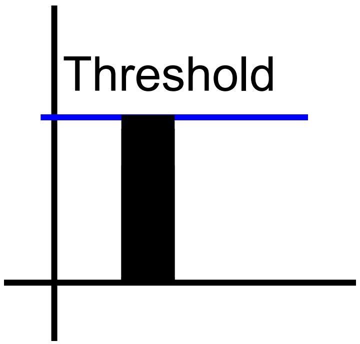


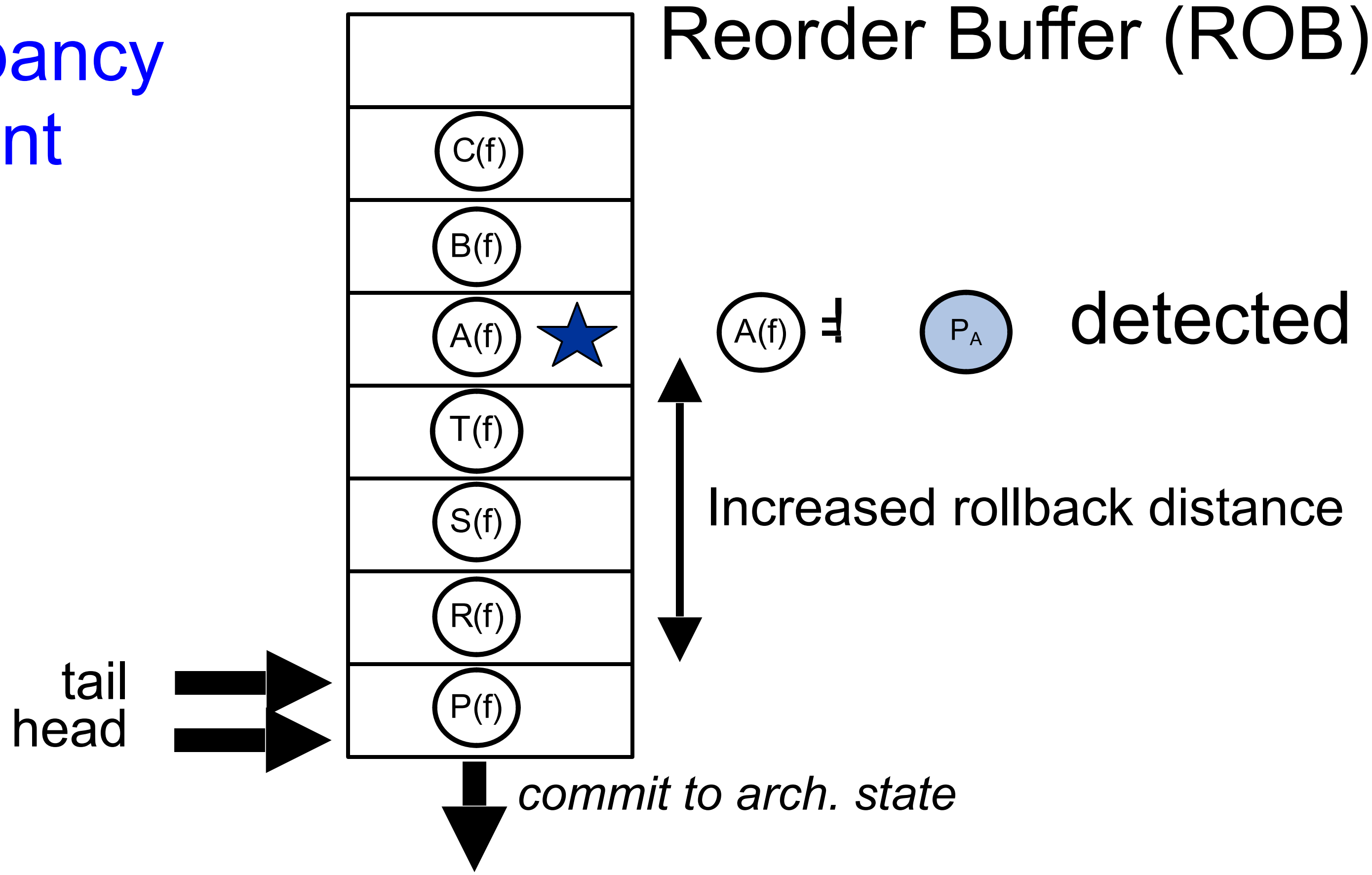
Flush to ROB head

Limited rollback distance:

- R-stream retires quickly – accelerated by leading A-stream

Enhancement: ROB occupancy management



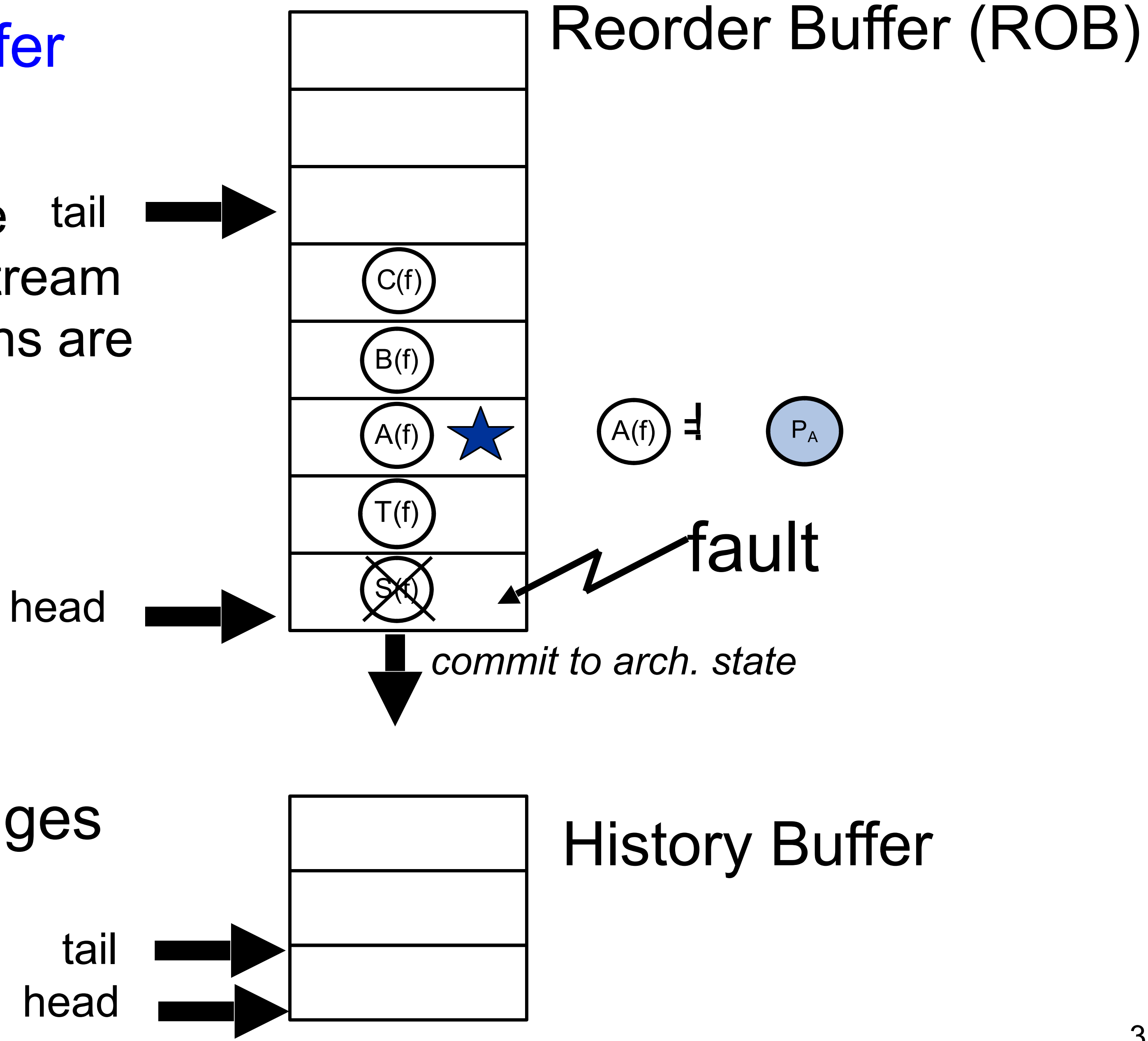


- Delay in retirement, hence performance hit

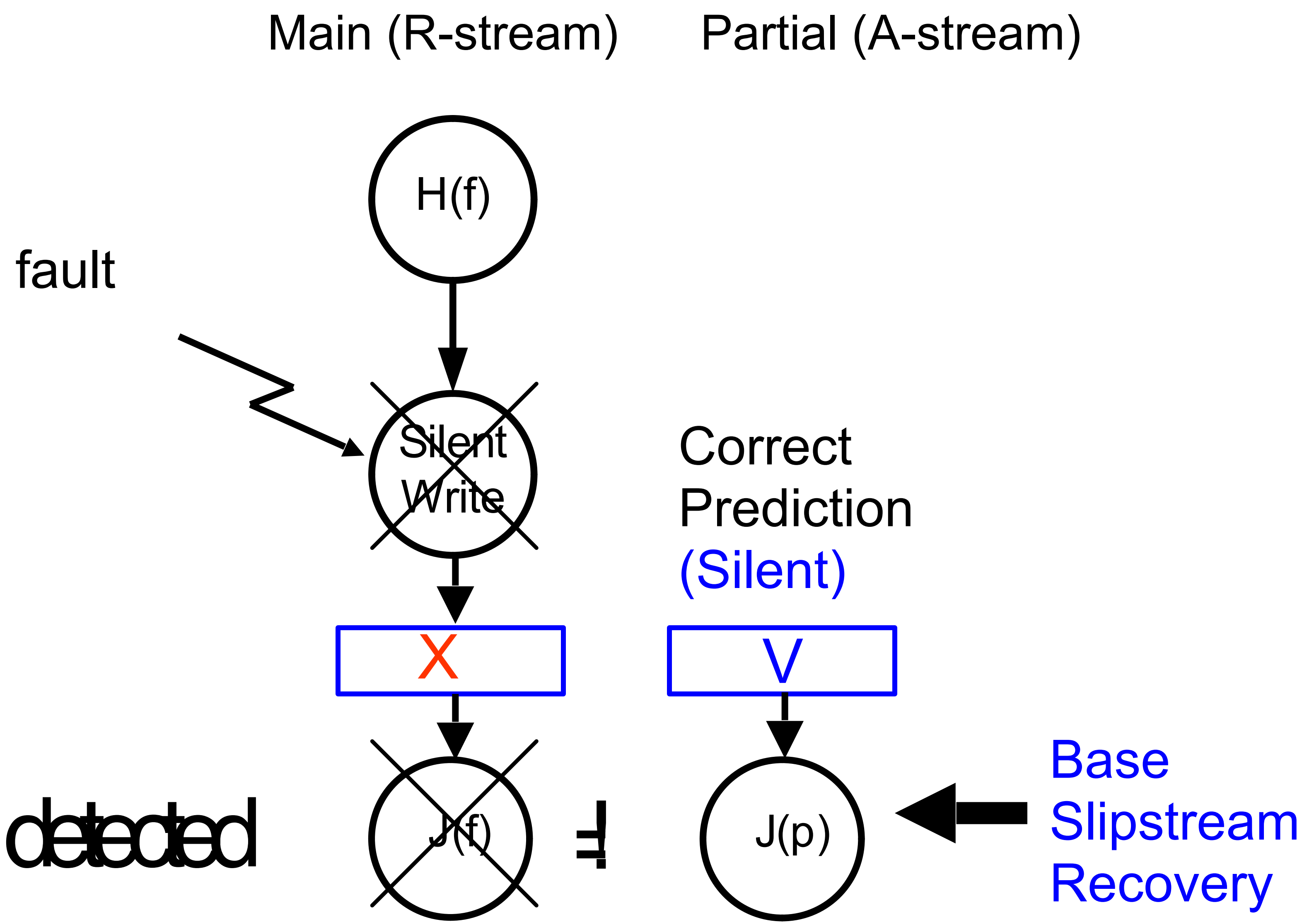
Enhancement: History Buffer

Performance friendly: R-stream mispredictions are rare

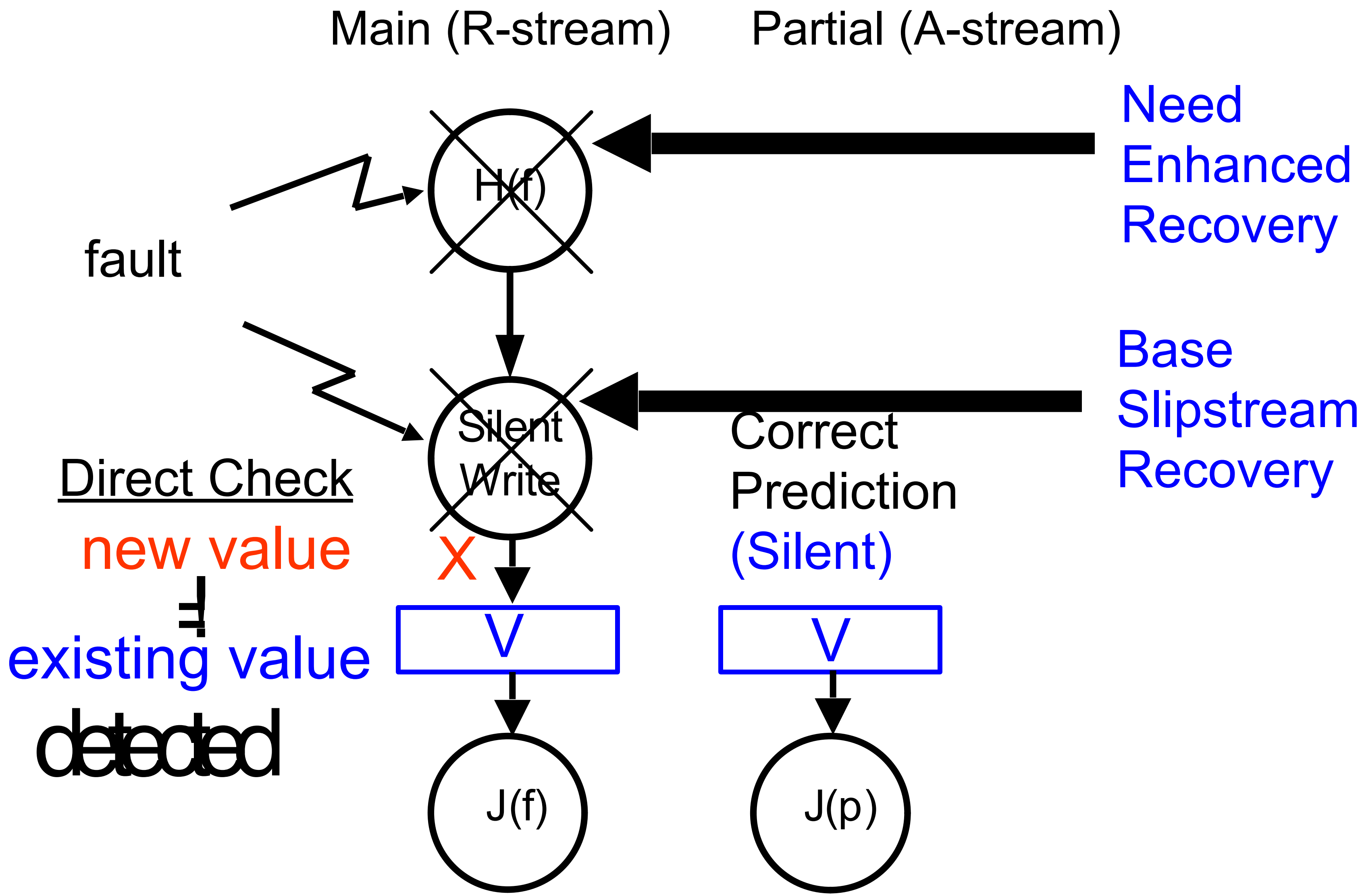
Undo changes



Indirect Check of Silent Write/Store



Direct Check of Silent Write/Store

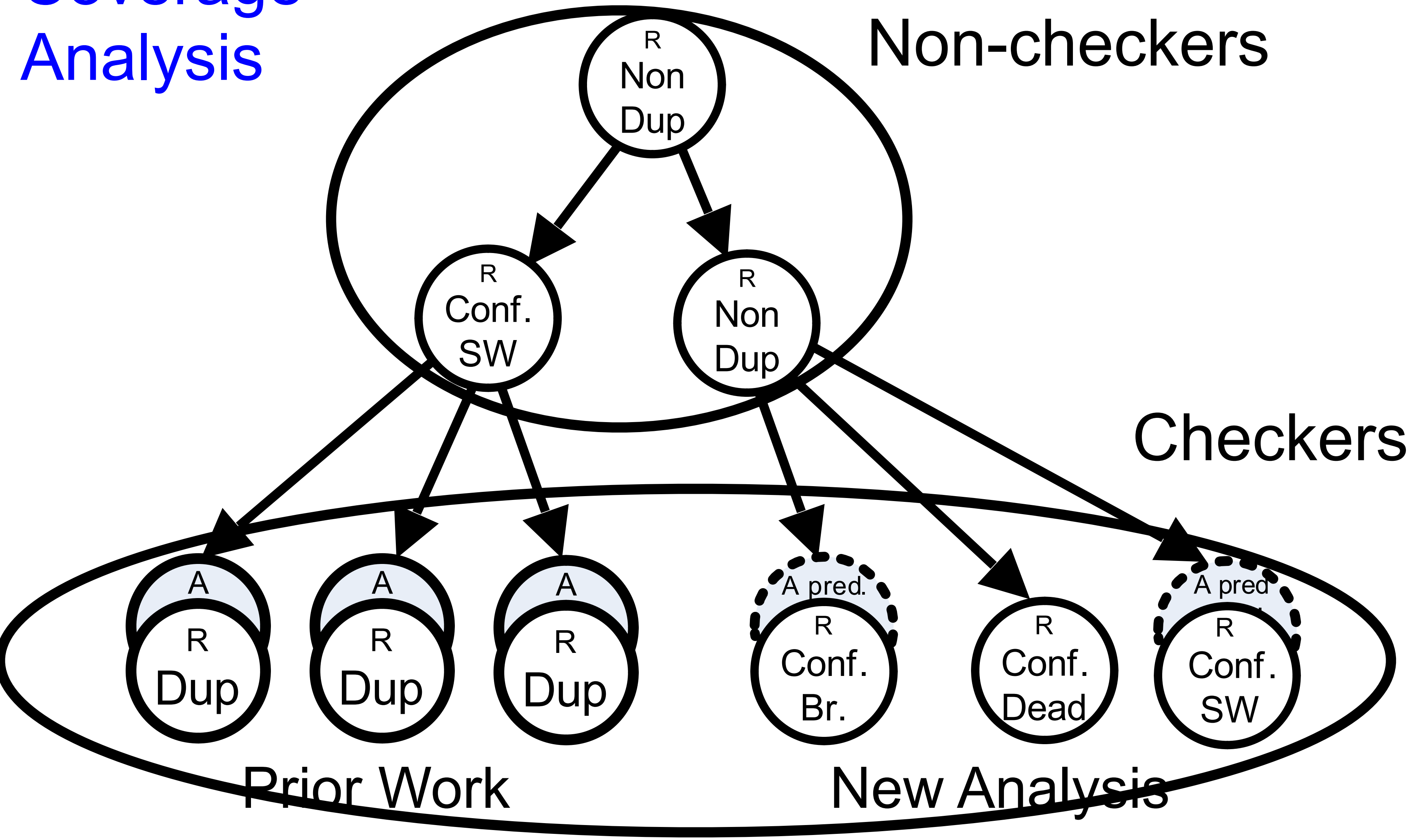


Novel Framework to Analyze Coverage

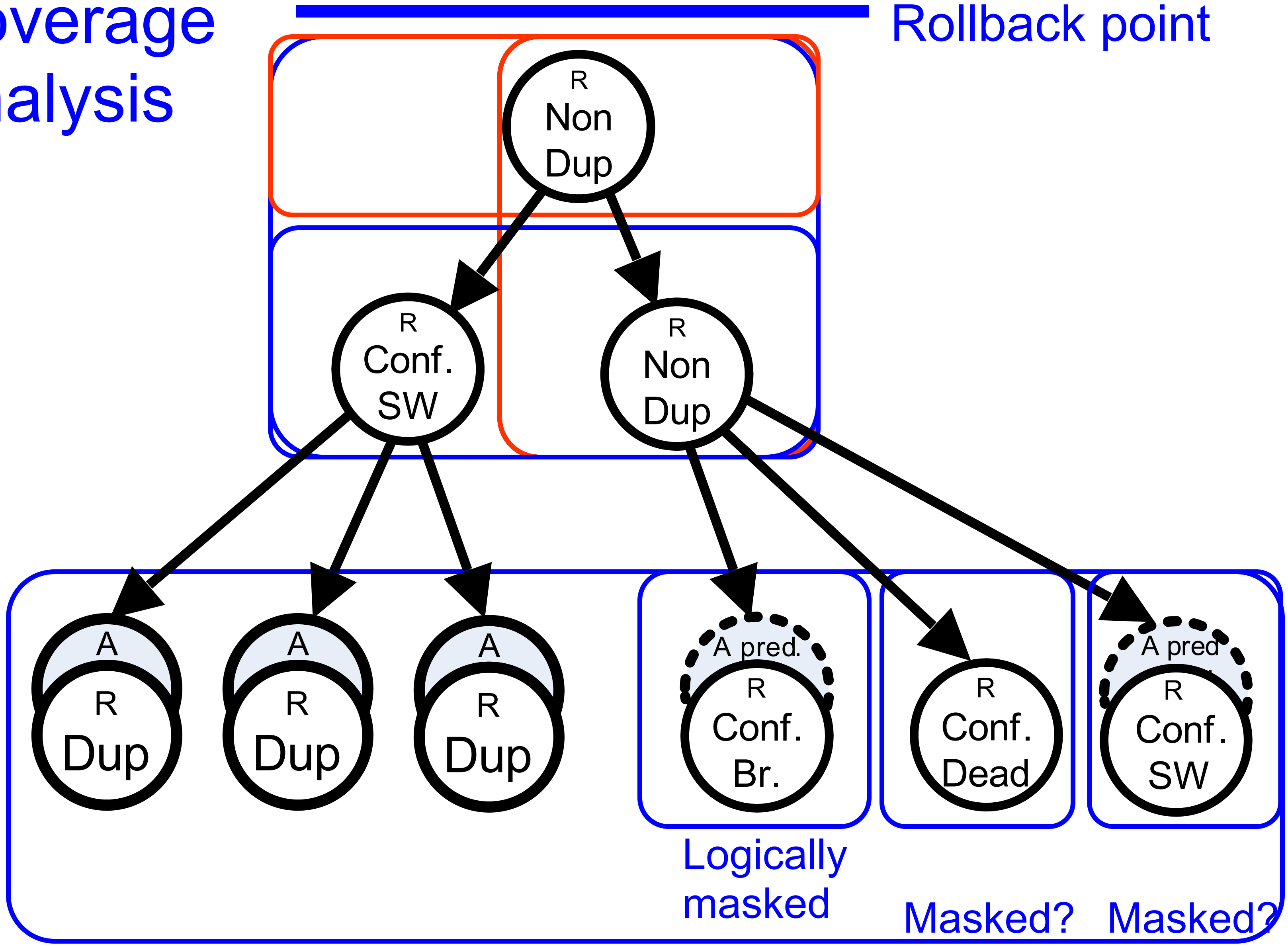
- Each instr. considered “candidate faulty”
- Coverage = # of instr. checked before committal to arch. state

- Mispredicted instr. and backward slices marked unchecked

Coverage Analysis



Coverage Analysis



Clarification

- Analysis framework is a coverage measurement tool
- Not in the actual hardware

Results: Microarchitecture models

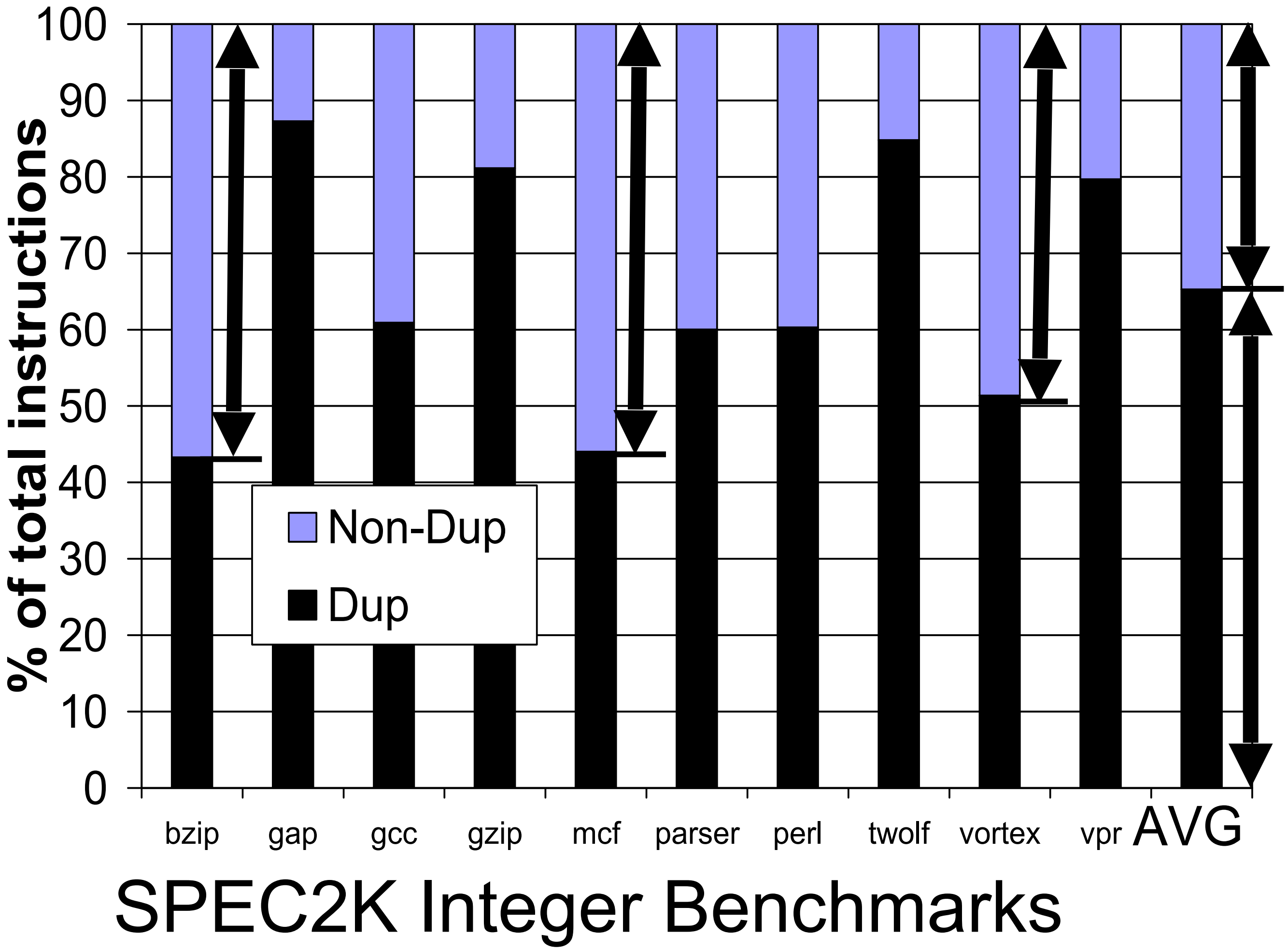
L1 I & D caches	64KB, 4-way, 64B line, LRU, L1hit = 1 cycle, L1miss/L2hit = 10 cycles
L2 unified cache	1MB, 8-way, 64B line, LRU, L1miss/L2miss = 100 cycles
superscalar core	dispatch/issue/retire bandwidth: 8 (4) reorder buffer (ROB): 256 (128) load/store queue: 64 (32) issue queue: 64 (32) cache ports (read/write): 4 (2)

Breakdown of Instructions

65% duplicated

35% non-duplicated

> 50% non-duplicated on some bench.

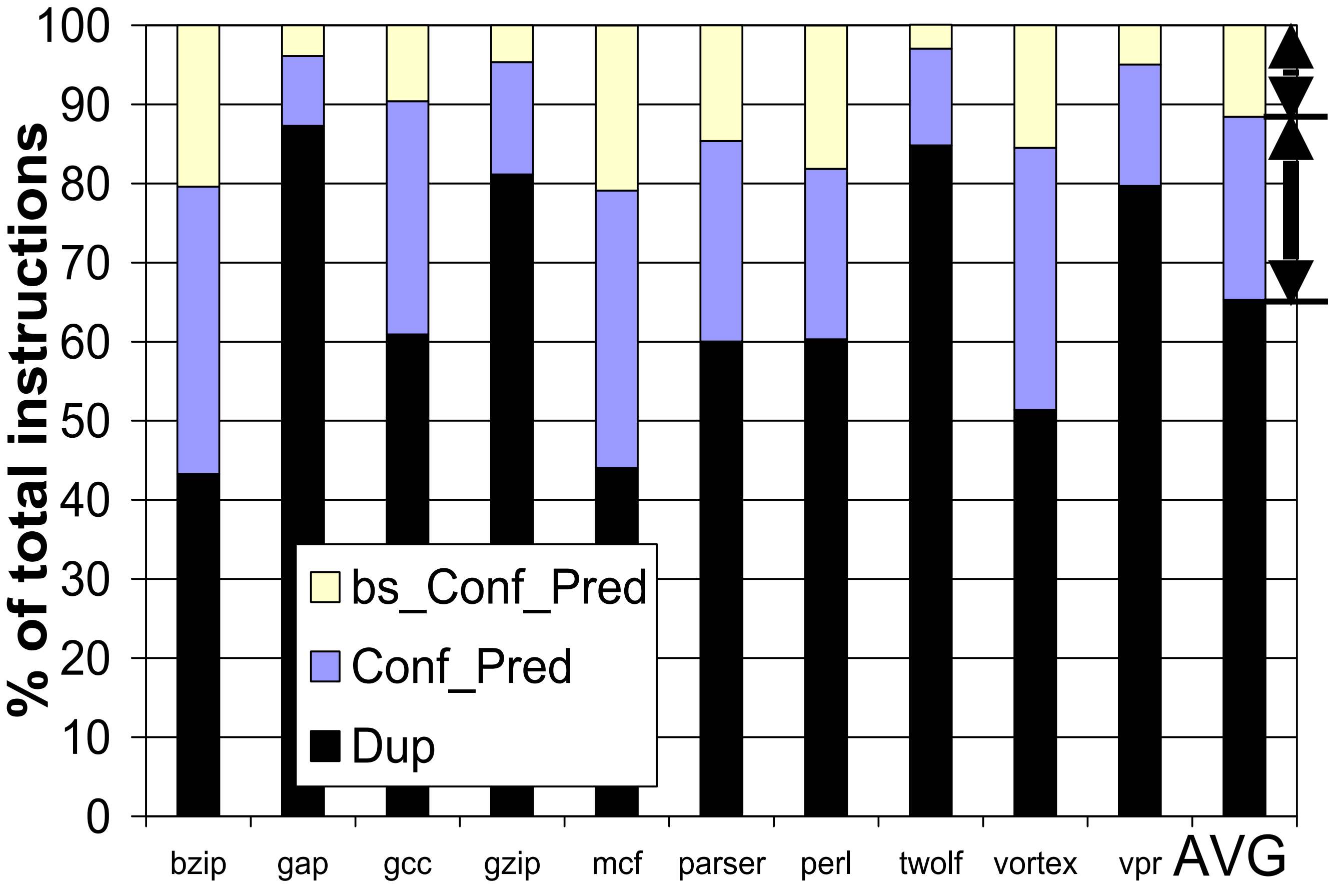


Breakdown of Instructions

23% confident predictions

12% backward slices of confident predictions

See paper for more detailed breakdown



Fault Coverage

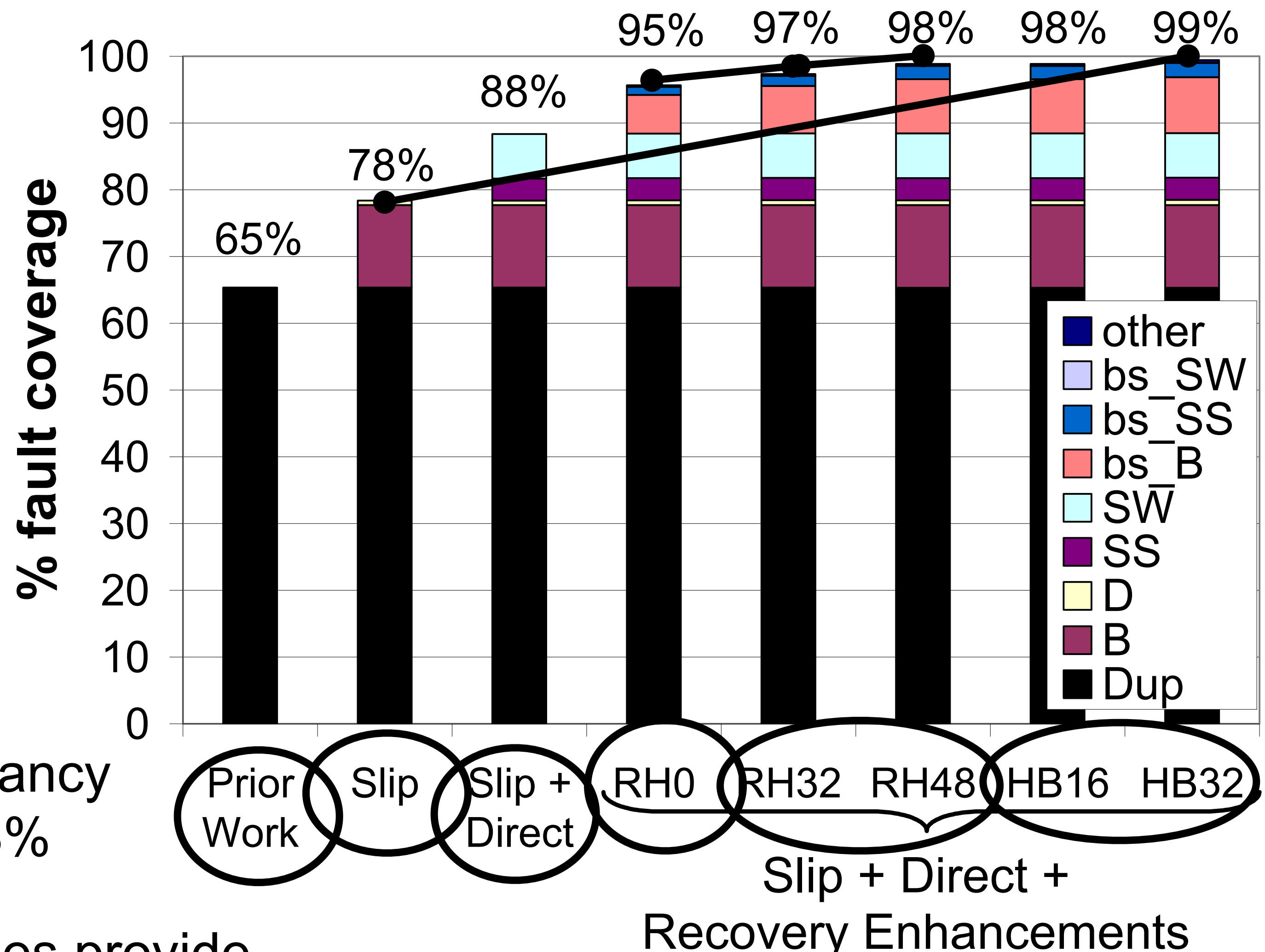
Prior Work (65%): Only dup. instr.

New result (78%): Correct confident predictions covered

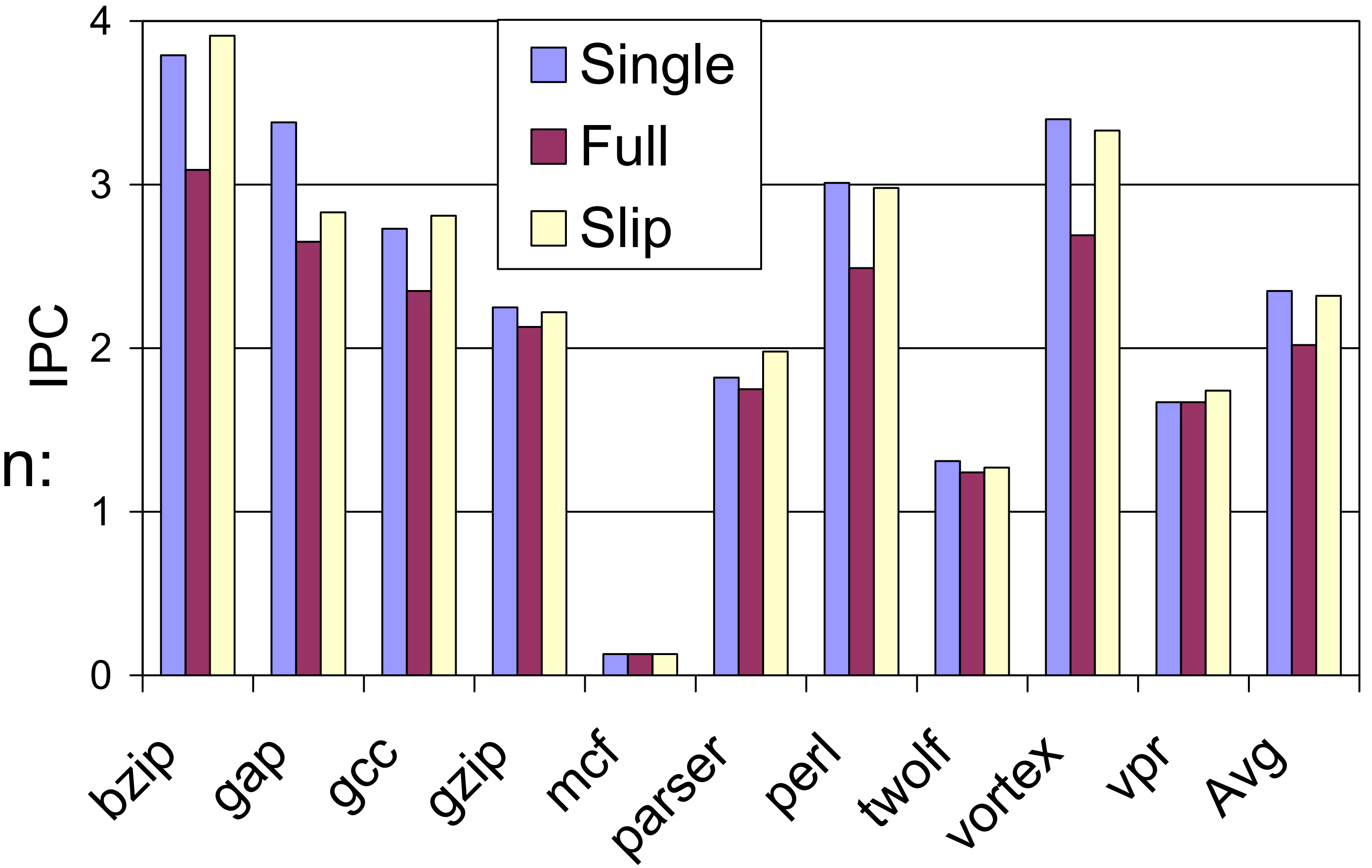
Direct silent write/ store checks improve coverage (88%)

ROB-head recovery improves with occupancy threshold: 95% to 98%

History buffer schemes provide high coverage (98% to 99%)



Slipstream Performance (SMT 8-wide)



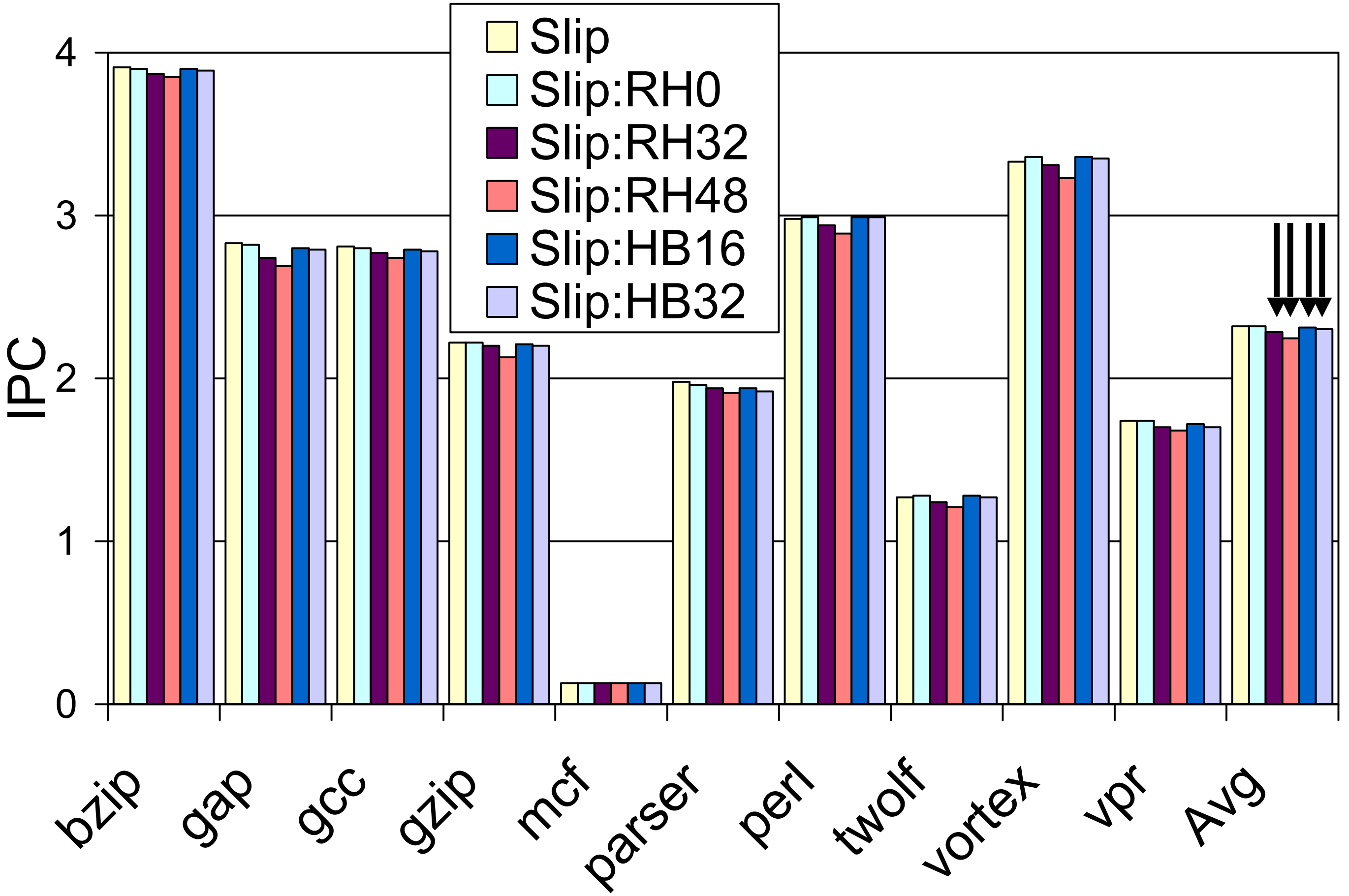
Average slowdown:

Full redundant execution:14%

Slipstream:1.3%

Performance Impact of Enhanced Recovery

- ROB-occupancy management delays retirement, causes slowdown (gradual decrease from RH0 to RH48)
- History buffer approach is performance friendly (negligible slowdown)



Conclusions

- Confident predictions can replace duplication
  – Slipstream case study : Redundant thread reduced by up to 57% while retaining near-100% coverage
- Prediction-based PRT offers a new avenue for efficient fault tolerant computing