Scalable Failure Recovery for Tree-based Overlay Networks

Dorian C. Arnold
University of Wisconsin

Paradyn/Condor Week
April 30 – May 3, 2007
Madison, WI

Overview

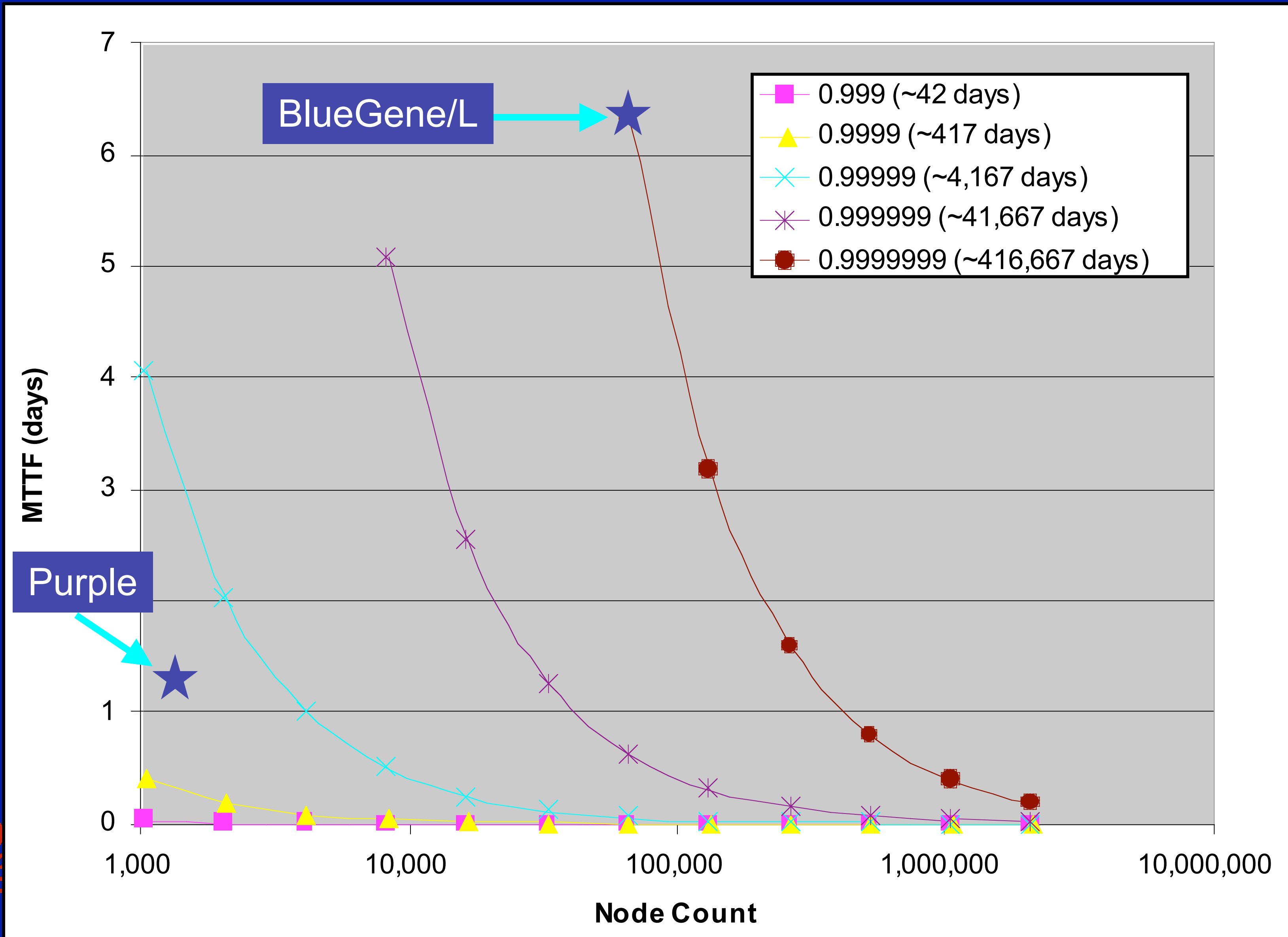
- Motivation
  - Address the likely frequent failures in extreme-scale systems
- State Compensation:
  - Tree-Based Overlay Networks (TBŌNs) failure recovery
  - Use surviving state to compensate for lost state
  - Leverage TBŌN properties
    - Inherent information redundancies
  - Weak data consistency model: convergent recovery
    - Final output stream converges to non-failure case
    - Intermediate output packets may differ
    - Preserves all output information

HPC System Trends

- 60% larger than 10^3 processors
- 10 systems larger than 10^4 processors

System	Location	Size	Time Frame
RoadRunner	LANL	$\sim 3.2 \times 10^4$	2008
Jaguar	ORNL	$\sim 4.2 \times 10^4$	2007
SunFire x64	TACC	$\sim 5.2 \times 10^4$	2007
Cray XT4	ORNL	$\sim 2 \times 10^5$	2008
BlueGene/P	ANL	$\sim 5 \times 10^5$	2008
BlueGene/Q	ANL/LLNL	$\sim 10^6$	2010-2012

Large Scale System Reliability



Current Reliability Approaches

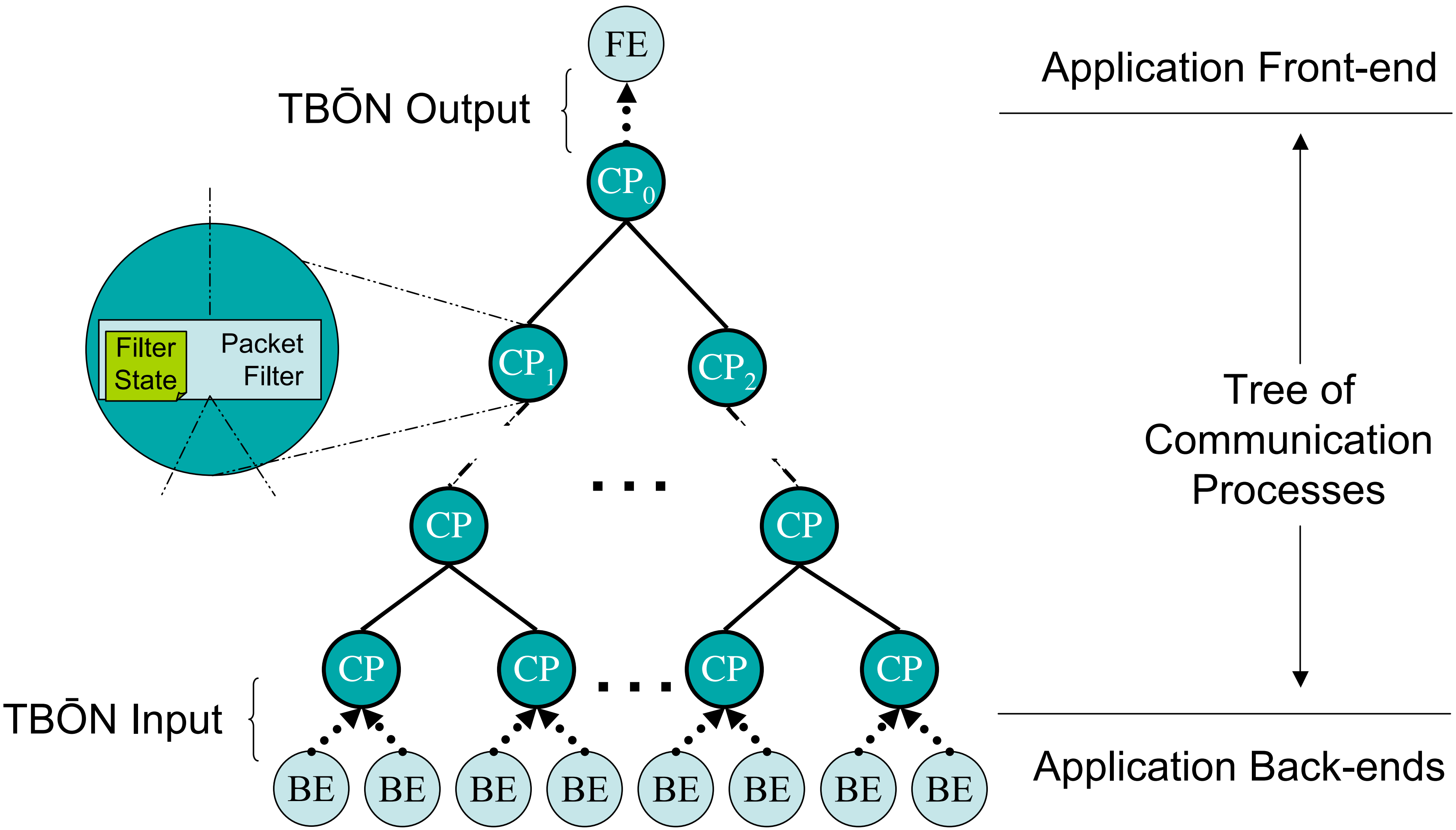
- Fail-over (hot backup)
  - Replace failed primary w/ backup replica
  - Extremely high overhead: 100% minimum!

- Rollback recovery
  - Rollback to checkpoint after failure
  - May require dedicated resources and lead to overloaded network/storage resources

Our Approach: State Compensation

- Leverage inherent TBŌN redundancies
  - Avoid explicit replication
- No overhead during normal operation
- Rapid recovery
  - Limited process participation
- General recovery model
  - Applies to broad classes of computations

Background: TBŌN Model



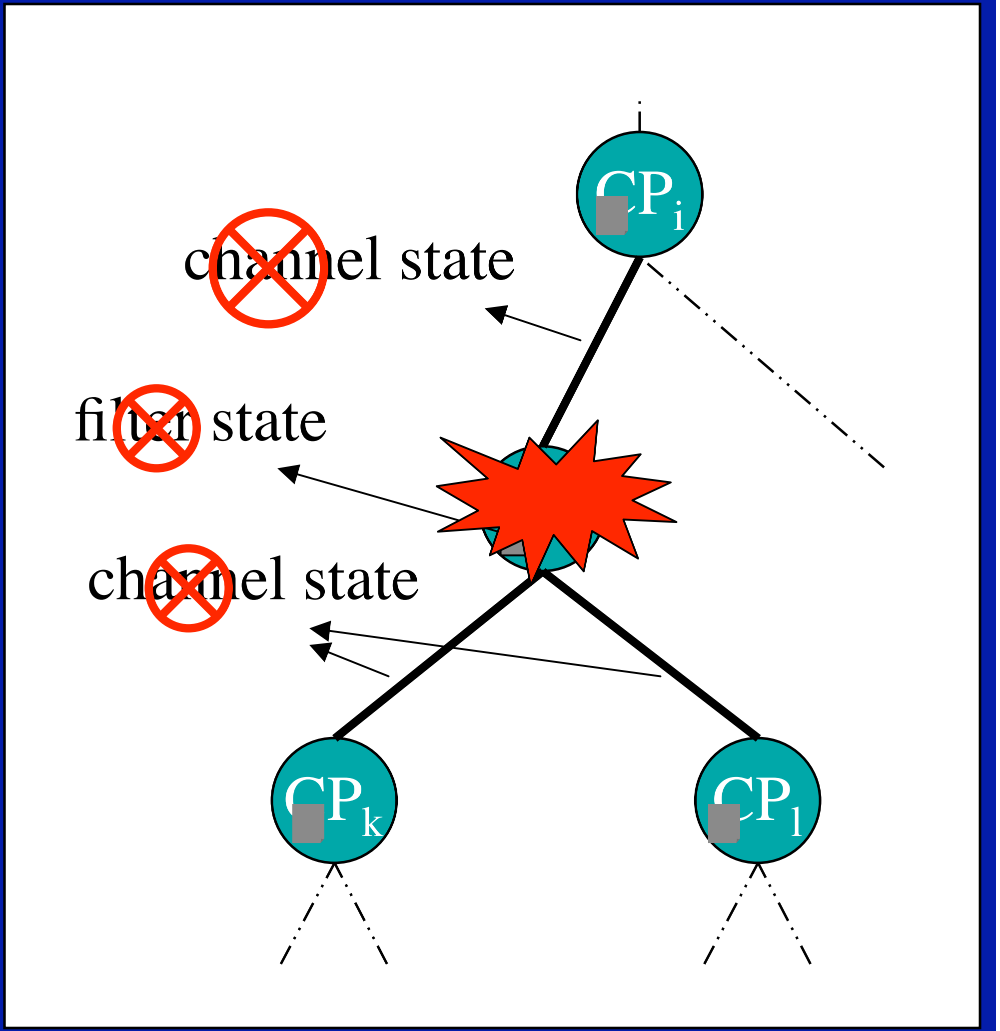
A Trip Down Theory Lane

- Formal treatment provides confidence in recovery model
  - Reason about semantics before/after recovery
    - Recovery model doesn't change computation
  - Prove algorithmic soundness
  - Understand recovery model characteristics

- System reliability cannot depend upon intuition and ad-hoc reasoning

Theory Overview

- TBŌN end-to-end argument: output only depends on state at the end-points
- Can recover from lost of any internal filter and channel states

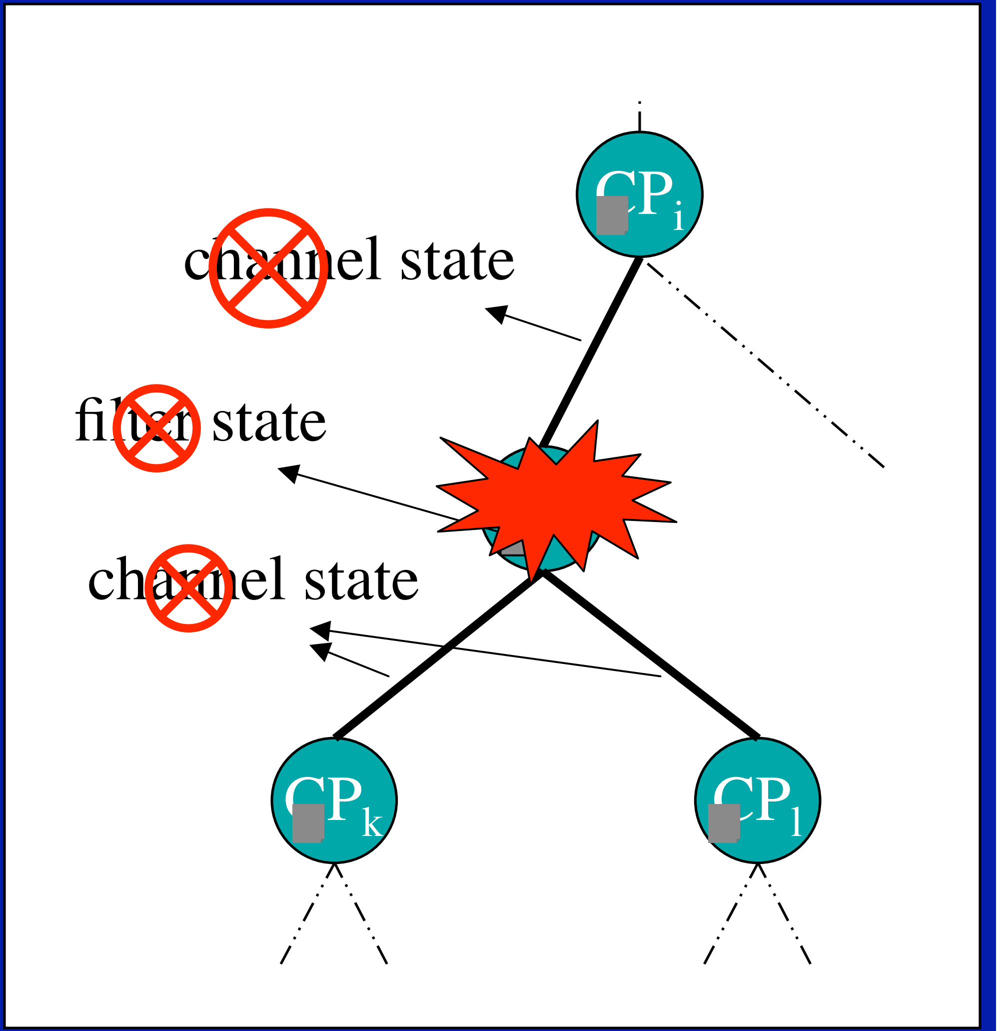


Theory Overview (cont'd)

TBŌN Output Theorem
Output depends only on channel states and root filter state

All-encompassing Leaf State Theorem
State at leaves subsume channel state (all state throughout TBŌN)

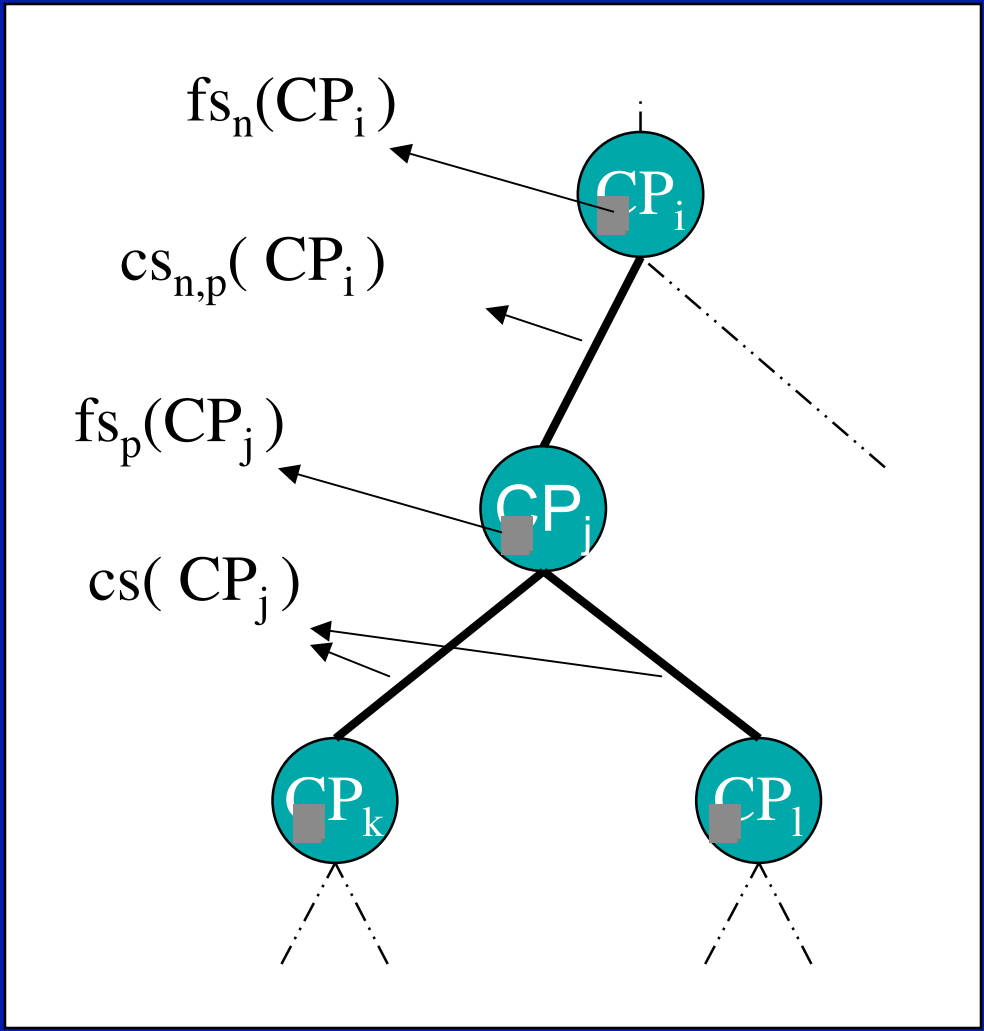
Result: only need leaf state to recover from root/internal failures



Theory Overview (cont'd)

- TBON Output Theorem
- All-encompassing Leaf State Theorem
  - Builds on Inherent Redundancy Theorem

Background: Notation



Background: Data Aggregation

Filter function:

$$f\,(in_n(CP_i); f\,s_n(CP_i))\ !\ \ f\,out_n(CP_i); f\,s_{n+1}(CP_i)g$$

Packets from input channels

Current filter state

Output packet

Updated filter state

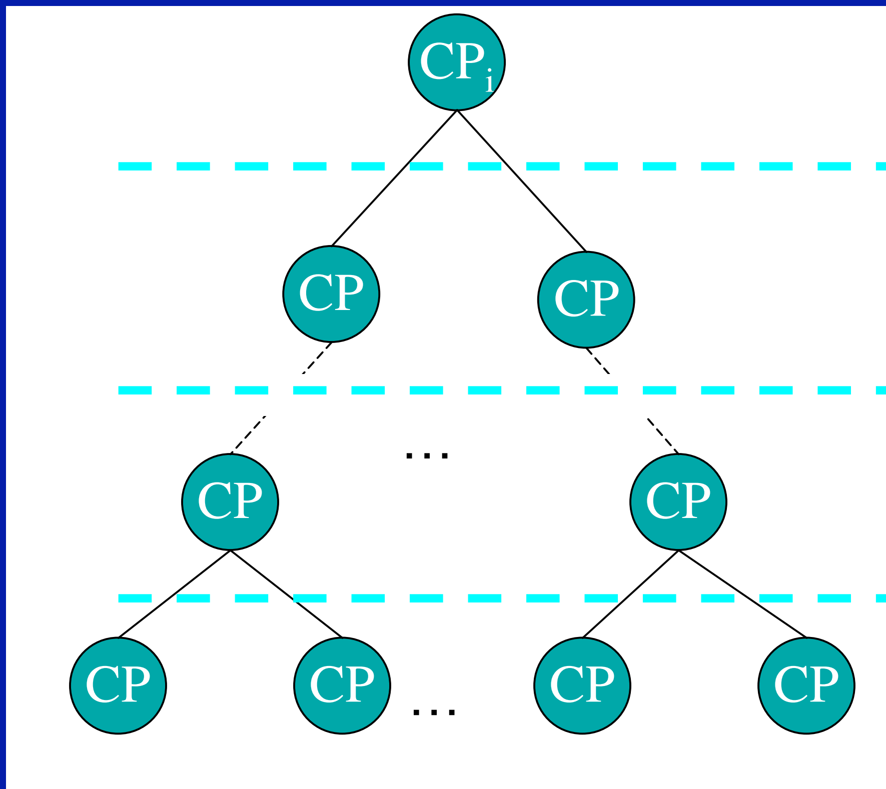
Background: Filter Function

- Built on state join and difference operators
- State join operator,
  - Update current state by merging inputs

$$in_n(CP_i)\ t\ fs_n(CP_i)\ !\ fs_{n+1}(CP_i)$$

  - Commutative: $a\,t\,b = b\,t\,a$
  - Associative: $(a\,t\,b)\,t\,c = a\,t\,(b\,t\,c)$
  - Idempotent: $a\,t\,a = a$

Background: Descendant Notation



$desc^0(CP_i)$

$desc^1(CP_i)$

$desc^{k-1}(CP_i)$

$desc^k(CP_i)$

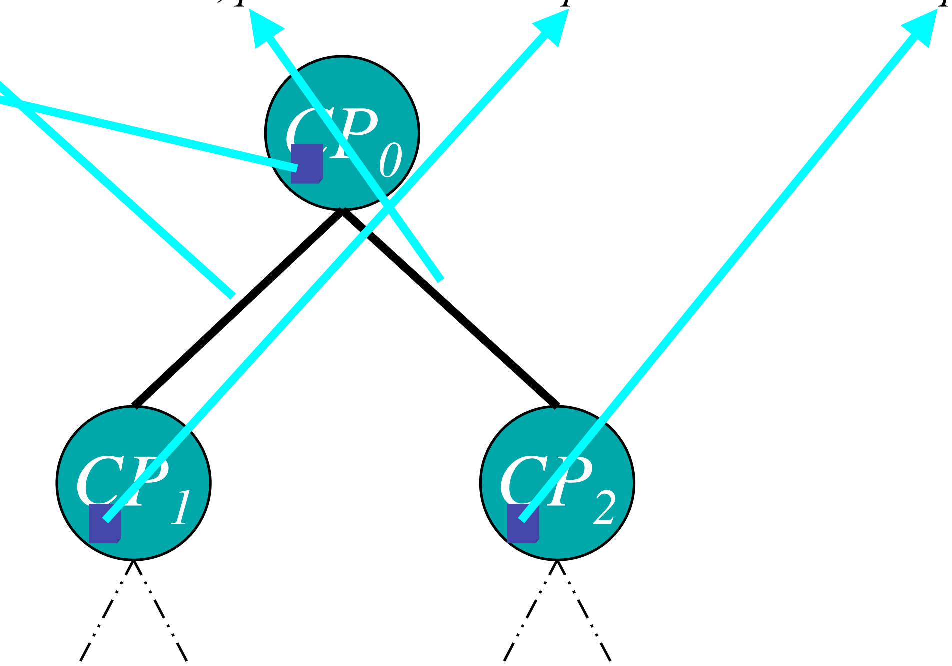
$fs(\ desc^k(CP_i)\ )$: join of filter states of specified processes

$cs(\ desc^k(CP_i)\ )$: join of channel states of specified processes

TBŌN Properties: Inherent Redundancy Theorem

The join of a CP's filter state with its pending channel state equals the join of the CP's children's filter states.

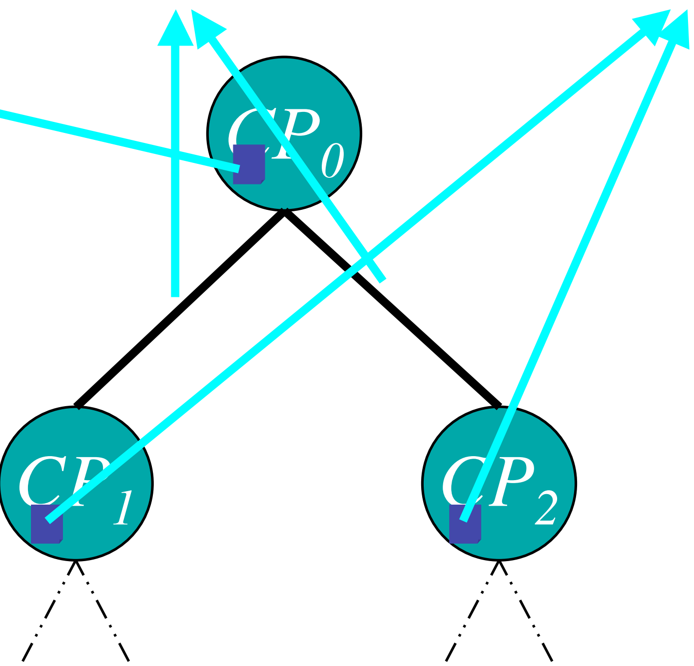
$$fs_n(CP_0) \sqcup cs_{n,p}(CP_0) \sqcup cs_{n,q}(CP_0) = fs_p(CP_1) \sqcup fs_q(CP_2)$$



TBŌN Properties: Inherent Redundancy Theorem

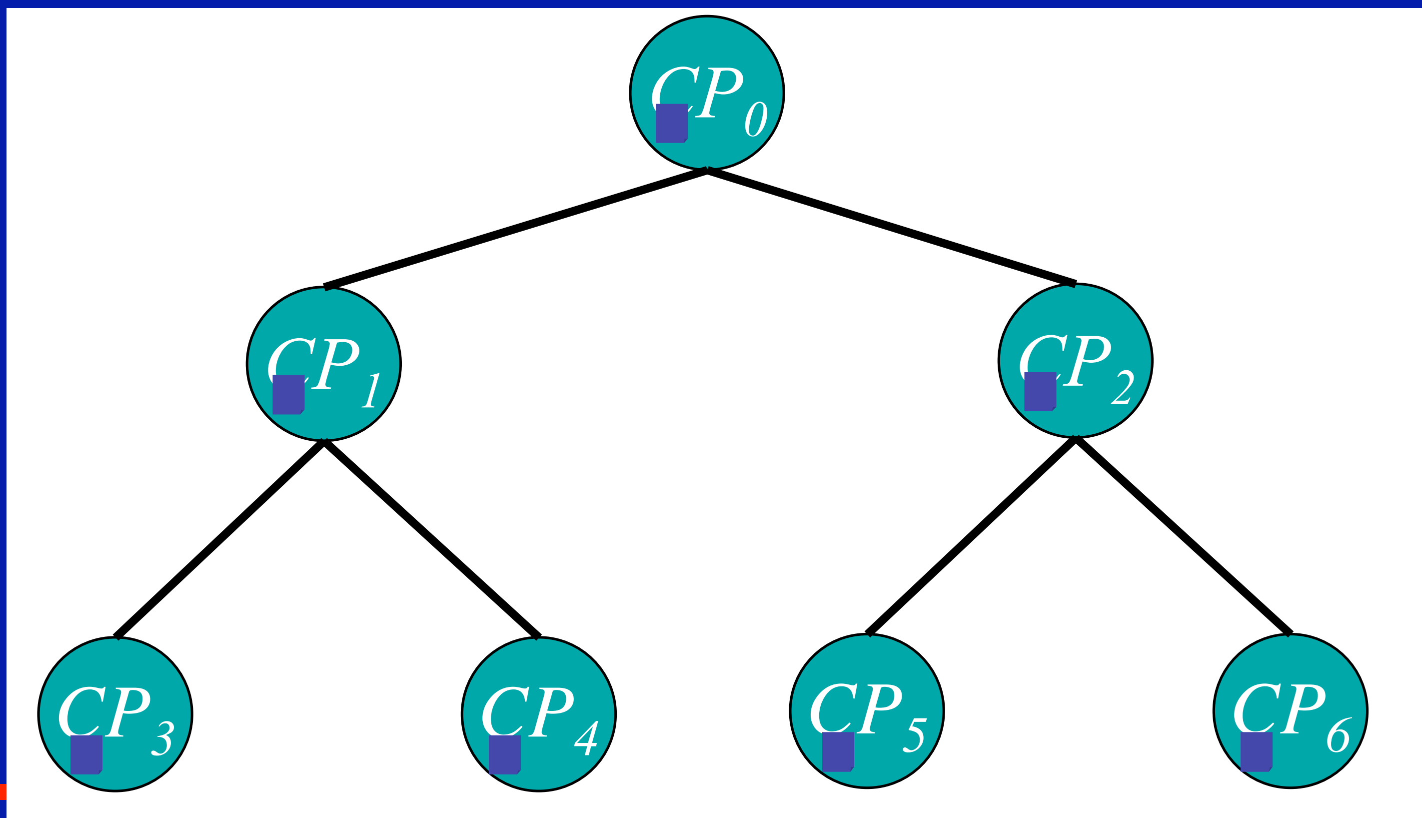
The join of a CP's filter state with its pending channel state equals the join of the CP's children's filter states.

$fs(\,desc^0(CP_0))$ $\quad\quad$ $cs(\,desc^0(CP_0)) \;=\; fs(\,desc^1(CP_0))$



TBŌN Properties:
All-encompassing Leaf State Theorem

The join of the states from a sub-tree's leaves equals the join of the states at the sub-tree's root and all in-flight data



TBŌN Properties:
All-encompassing Leaf State Theorem

The join of the states from a sub-tree's leaves equals the join of the states at the sub-tree's root and all in-flight data

From Inherent Redundancy Theorem:

$$fs(desc^{1}(CP_0)) = fs(desc^{0}(CP_0)) \sqcup cs(desc^{0}(CP_0))$$

$$fs(desc^{2}(CP_0)) = fs(desc^{1}(CP_0)) \sqcup cs(desc^{1}(CP_0))$$

$$\vdots$$

$$fs(desc^{k}(CP_0)) = fs(desc^{k-1}(CP_0)) \sqcup cs(desc^{k-1}(CP_0))$$

$$fs(desc^{k}(CP_0)) = fs(CP_0) \sqcup cs(desc^{0}(CP_0)) \sqcup \ldots \sqcup cs(desc^{k-1}(CP_0))$$

State Composition

- Motivated by previous theory
  - State at leaves of a sub-tree subsume state throughout the higher levels
- Compose state below failure zones to compensate for lost state
- Addresses root and internal failures
- State decomposition for leaf failures
  - Generate child state from parent and sibling's

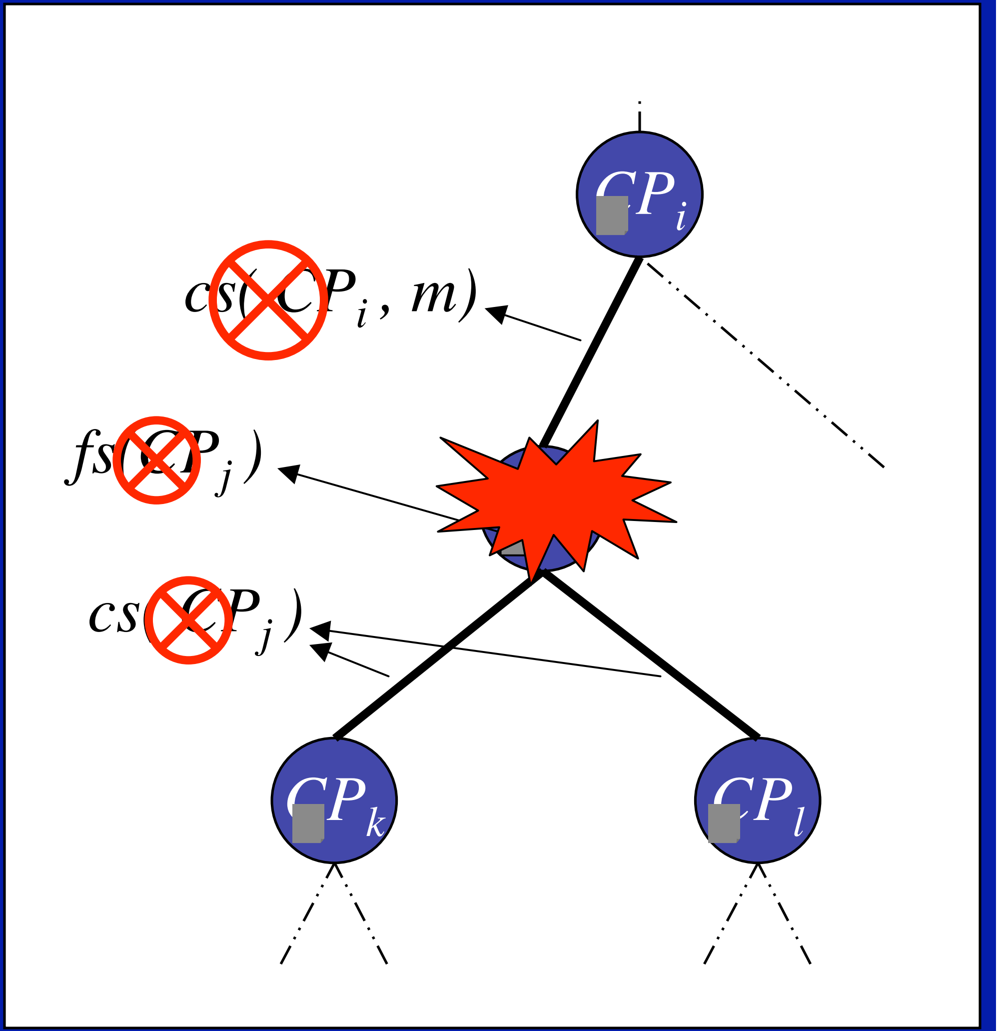
State Composition

If CP_j fails, all state associated with CP_j is lost

TBŌN Output Theorem:
Output depends only on channel states and root filter state

All-encompassing Leaf State Theorem:
State at leaves subsume channel state (all state throughout TBŌN)

Therefore, leaf states can replace lost channel state without changing computation's semantics



State Composition Algorithm

```
if detect child failure
    remove failed child from input list
    resume filtering from non-failed children
endif

if detect parent failure
    do
        determine/connect to new parent
    while failure to connect

    propagate filter state to new parent
endif
```

Summary: Theory can be Good!

- Allows us to make recovery guarantees
- Yields sensible, understandable results
- Better-informed implementation
  - What needs to be implemented
  - What does not need to be implemented

References

- Arnold and Miller, "State Compensation: A Scalable Failure Recovery Model for Tree-based Overlay Networks", UW-CS Technical Report, February 2007.

- Other papers and software:
  http://www.paradyn/org/mrnet

Bonus Slides!

State Composition Performance

recovery latency = (connection establishment £ max adoptees)+ output overhead

LAN connection establishment: ~1 millisecond

Failure Model

- Fail-stop
- Multiple, simultaneous failures
  - Failure zones: regions of contiguous failure
- Application process failures
  - May view as sequential data sources/sink
  - Amenable to basic reliability mechanisms
    - Simple restart, sequential checkpointing