

Tree-based Overlay Networks and Scalability Strategies for Petascale Tools

Dorian Arnold
(with Phil Roth and Bart Miller)
University of Wisconsin

IBM Petascale Tools Strategy
Workshop
May 3-4, 2005
Yorktown Heights, NY



The March toward Petascale

- Processor Statistics from Top500 List:
 - 7974: Top ten average
 - 18%: ≥ 1024
 - 59%: clusters
 - 8192: largest cluster
- In 2005:
 - 65,536x2 processor system

Clusters and MPPs w/ 10^4 - 10^5 processors will soon be commonplace.

Challenges of Petascale Systems

#1. Performance

- Efficient group communication
- Scalable information dissemination
- Scalable data collection and analyses

#2. Reliability

- Persistent state recovery
- Reliable group communication
- Failure detection

Preview

- Performance at scale
 - MRNet Overview
- Reliability at scale (as time permits)
 - Fault-tolerant strategies for tree-based overlay networks (TBONs)

Tree-based overlay networks
are a key to “petascalable” tools.

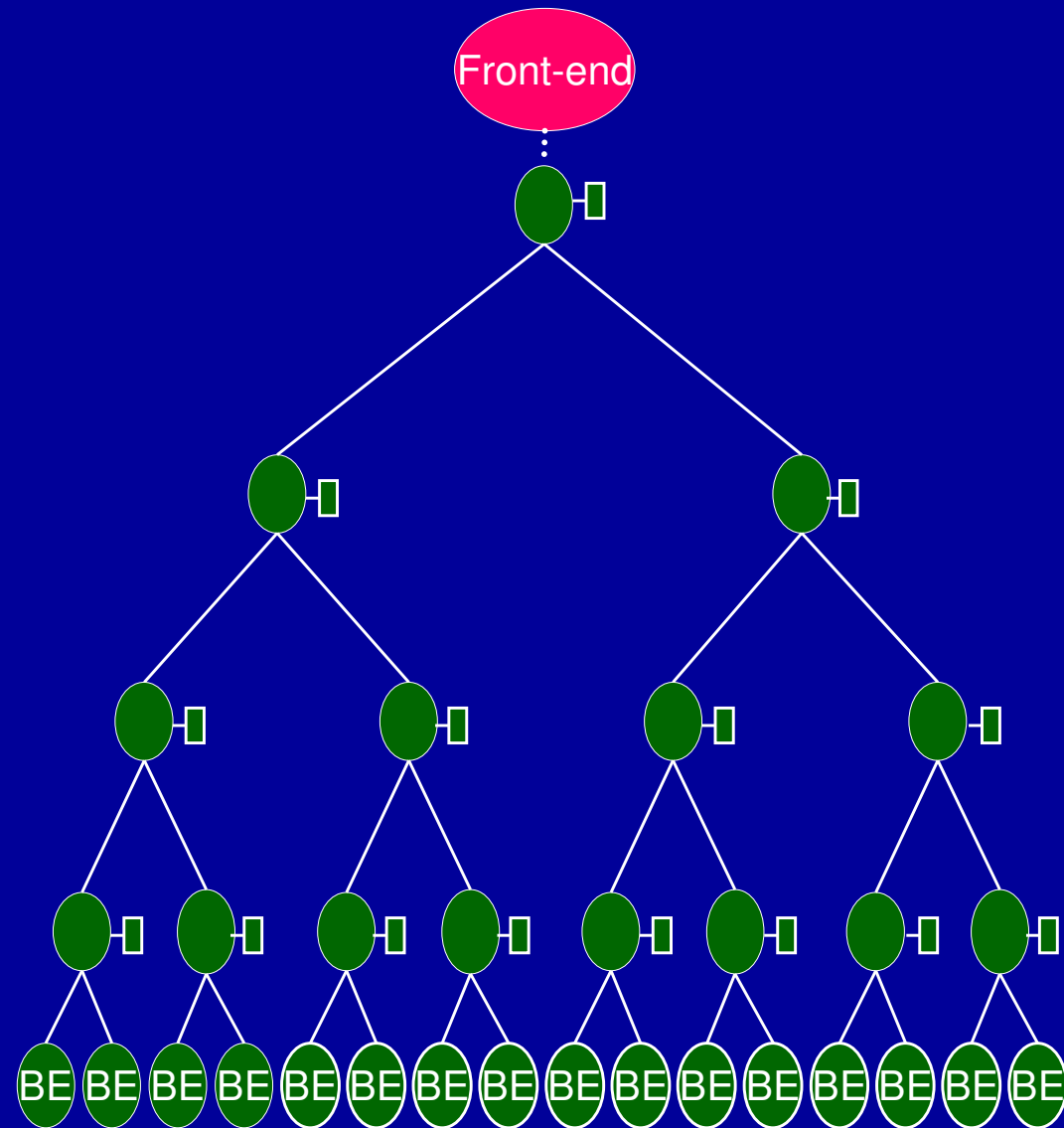
Petascale Challenge #1: Performance

- MRNet: Multicast/Reduction Overlay Network
 - T-BON for scalable, efficient group communications and data analyses
 - Scalable multicast
 - Scalable reduction
 - In-network data aggregation

MRNet Components and Features

Front-end

- User-Interface, control, visualization, ...



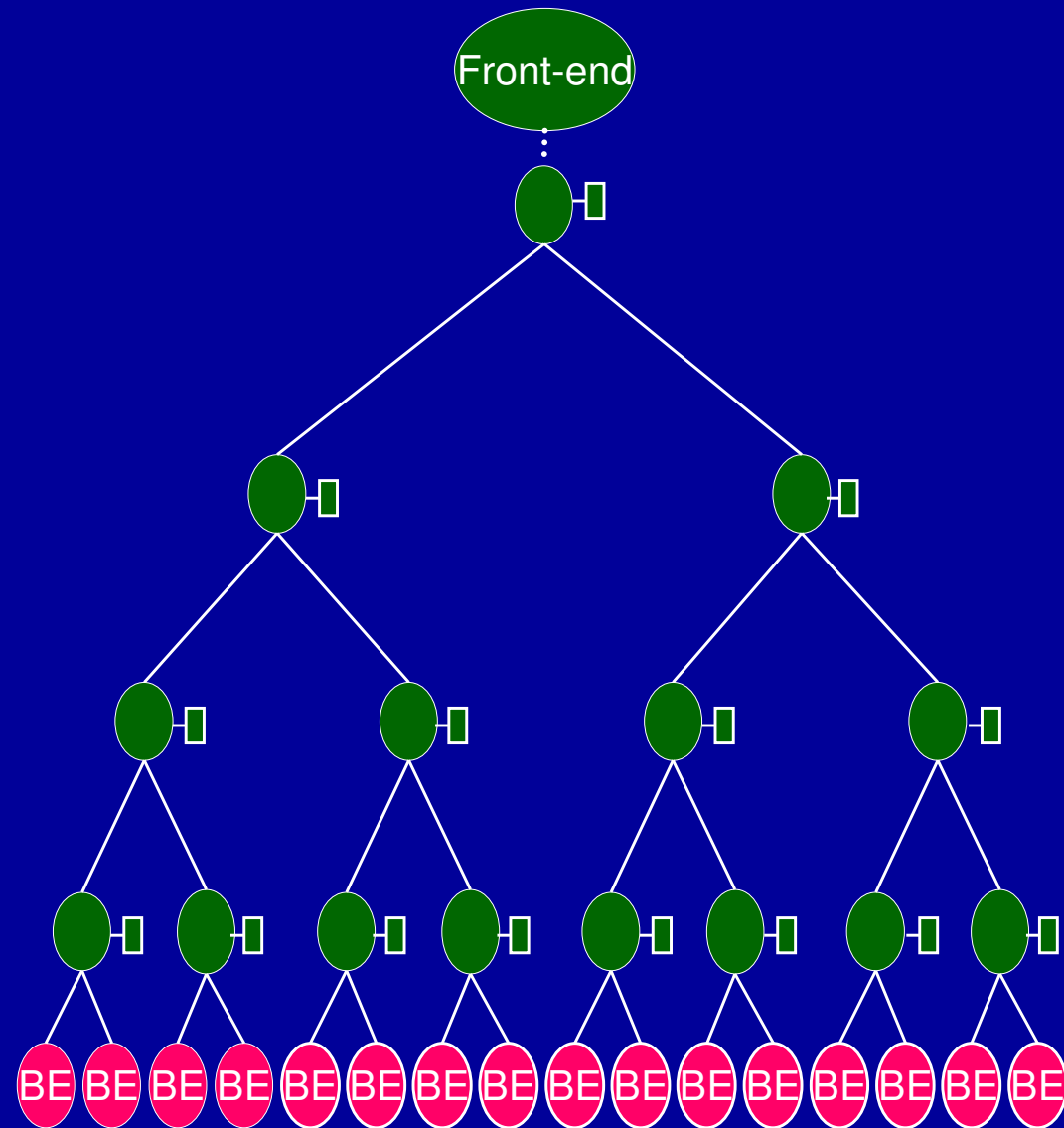
MRNet Components and Features

Front-end

- User-Interface, control, visualization, ...

Back-end

- Tool daemon, monitoring, profiling, ...



MRNet Components and Features

Front-end

- User-Interface, control, visualization, ...

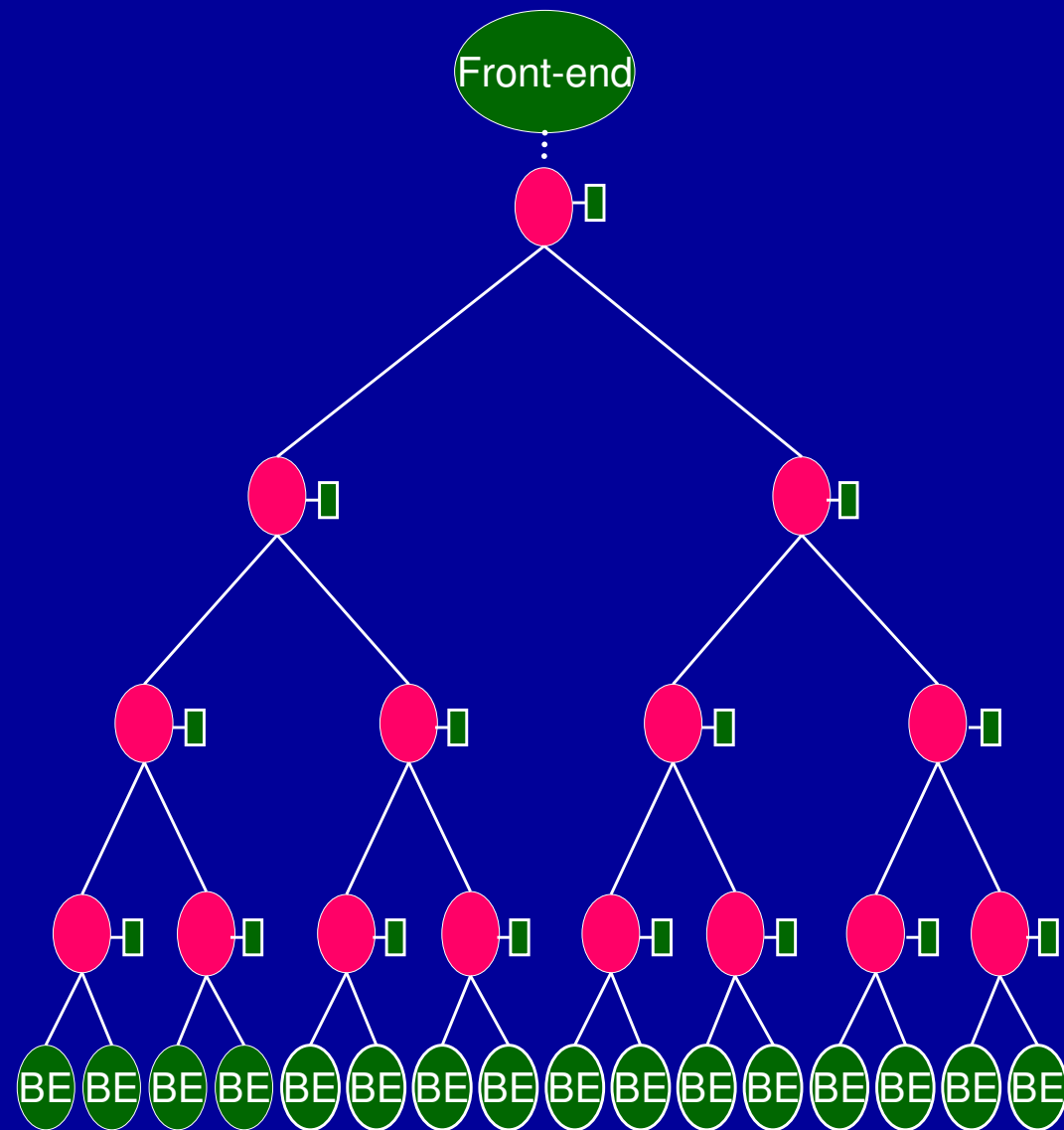
Back-end

- Tool daemon, monitoring, profiling, ...

Internal nodes

- Communication processes

- Flexible topology



MRNet Components and Features

Front-end

- User-Interface, control, visualization, ...

Back-end

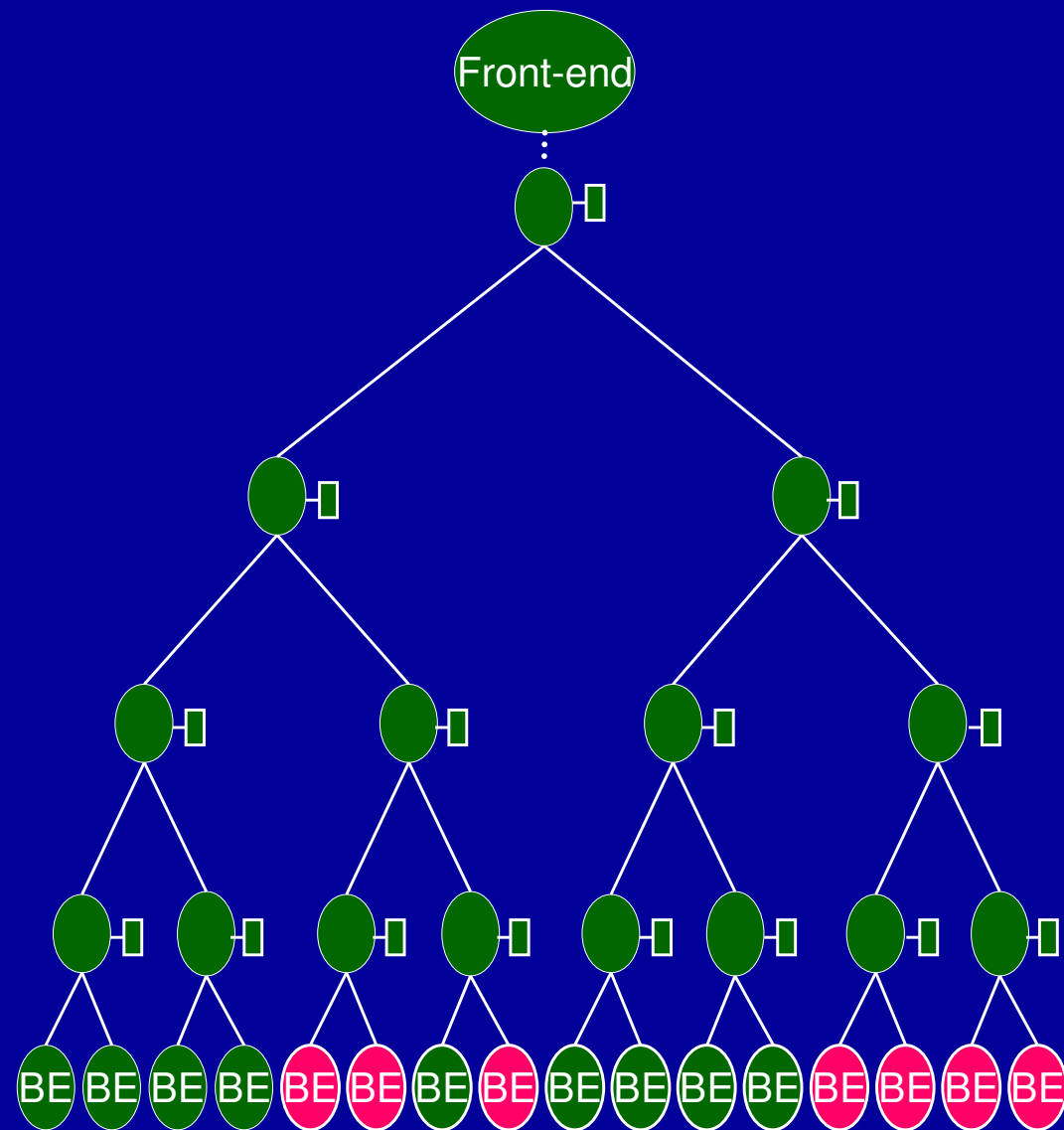
- Tool daemon, monitoring, profiling, ...

Internal nodes

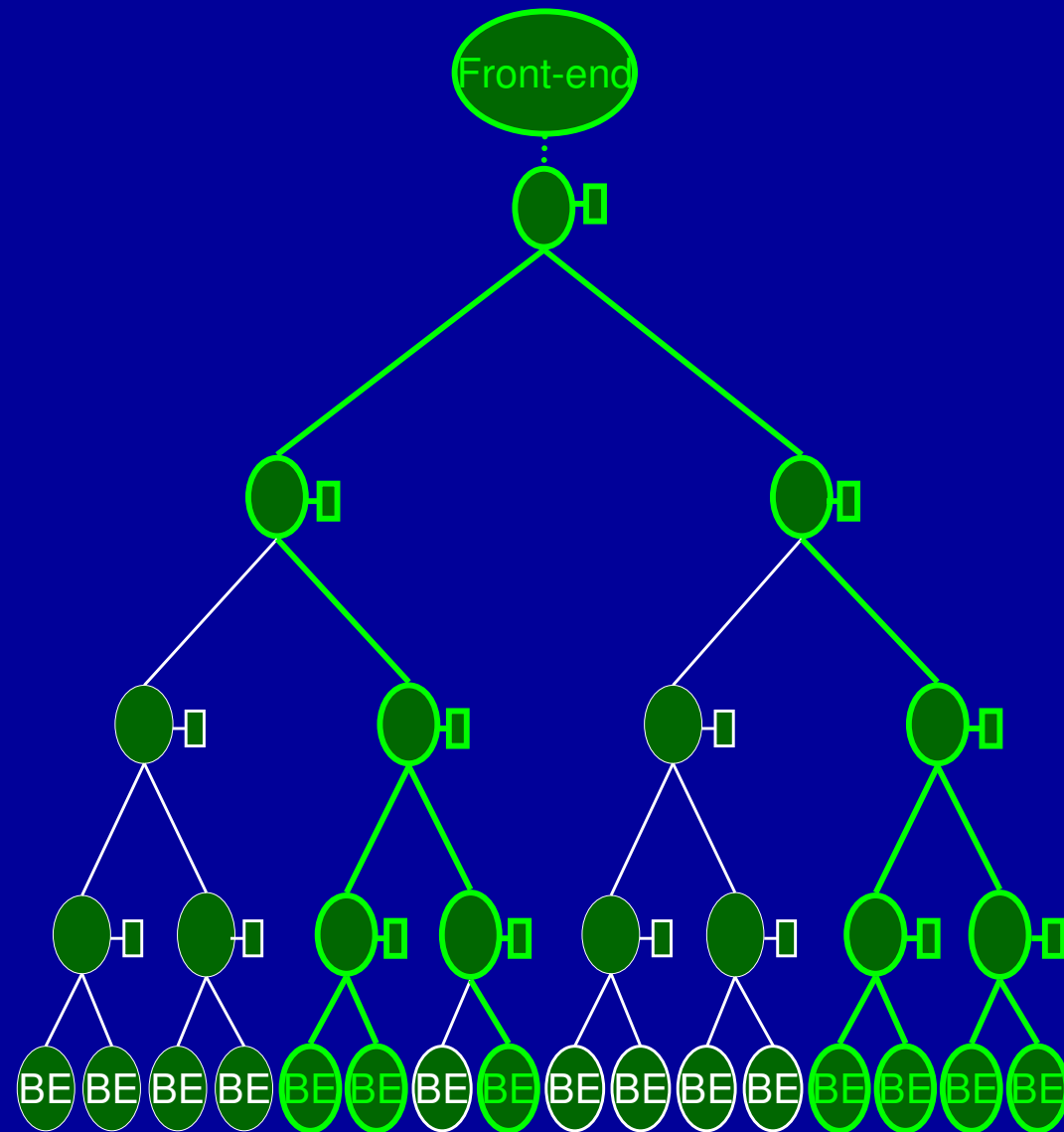
- Communication processes
- Flexible topology

Communicators

- Groups of endpoints



MRNet Components and Features



Front-end

- User-Interface, control, visualization, ...

Back-end

- Tool daemon, monitoring, profiling, ...

Internal nodes

- Communication processes
- Flexible topology

Communicators

- Groups of endpoints

Filters

- Operates on data flows
- Built-in or custom
- Transformation/synchronization

Streams

- Virtual communication channels

MRNet Interface

- Tools link with **libmrnet**, a library that exposes the MRNet C++ API.
- Abstractions include:
 - Network:
 - Initialize/shut-down network
 - Access network end-points
 - End-points
 - Communicators
 - Streams
 - Packets

MRNet Front-end Interface

```
1. front_end_main() {
2.     Network * net;
3.     Communicator * comm;
4.     Stream * stream;
5.     int result;

6.     net = new Network (config);
7.     comm = get_BroadcastCommunicator();
8.     stream = new Stream(comm, IMAX_FILT,
                          WAITFORALL_SYNC);
9.     stream->send("%s", "go");
10.    stream->recv("%d", result);
11. }
```

MRNet Back-end Interface

```
1. back_end_main(){
2.     Stream * stream;
3.     char s[128];

4.     Network::init_backend( );

5.     Stream::recv("%s", s, &stream);
6.     if(s == "go"){
7.         stream->send("%d", rand_int);
8.     }
9. }
```

MRNet Filter Interface

```
1. imax_filter(vector<Packet> packets_in,  
              vector<Packet> packets_out)  
   {  
     for( i=0; i<packets_in.size; i++){  
2.       result = max(result,  
                    packets[i].get_int());  
     }  
  
3.   Packet p("%d", result);  
  
4.   packets_out.pushback(p);  
   }
```

Petascale Challenge #2: Reliability

- Petascale systems must be reliable
- As systems get larger, expected time to system failure gets smaller
- Tools and applications must be able to tolerate large classes of failure

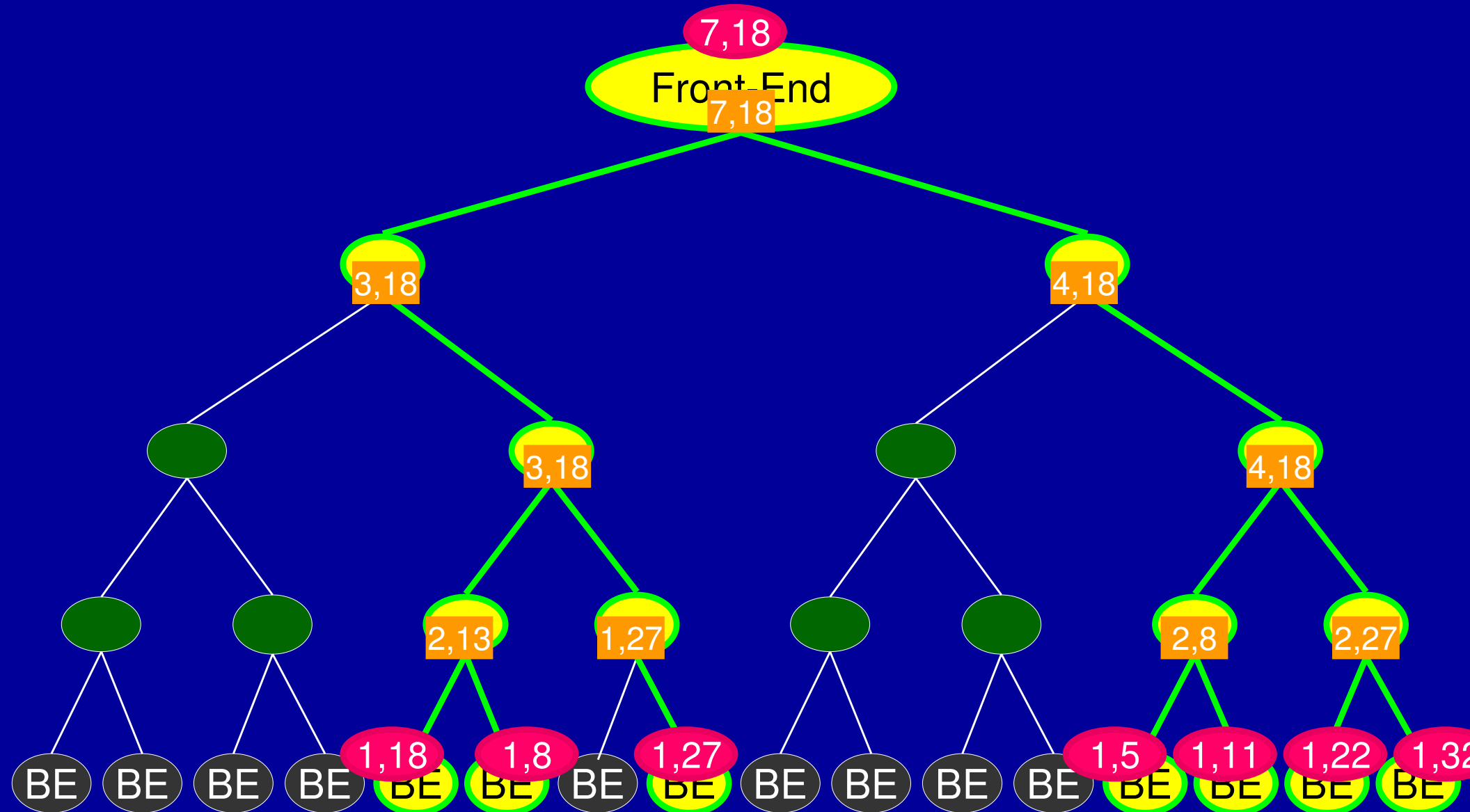
Main-Memory Implicit Checkpointing

- State recovery for non-transient failures
- Leverage **inherent redundancy** of stateful reduction networks
 - Based on **reversibility property** of reduction operations
- Use **volatile storage**
 - Reduces checkpoint latency
 - Checkpoint used to regenerate the state of **non-local failures**
- Establish **recovery clique**
 - Efficient, localized recovery semantics
- Assumes reliable transmission layer
 - All in-flight messages will eventually be received by destination processes

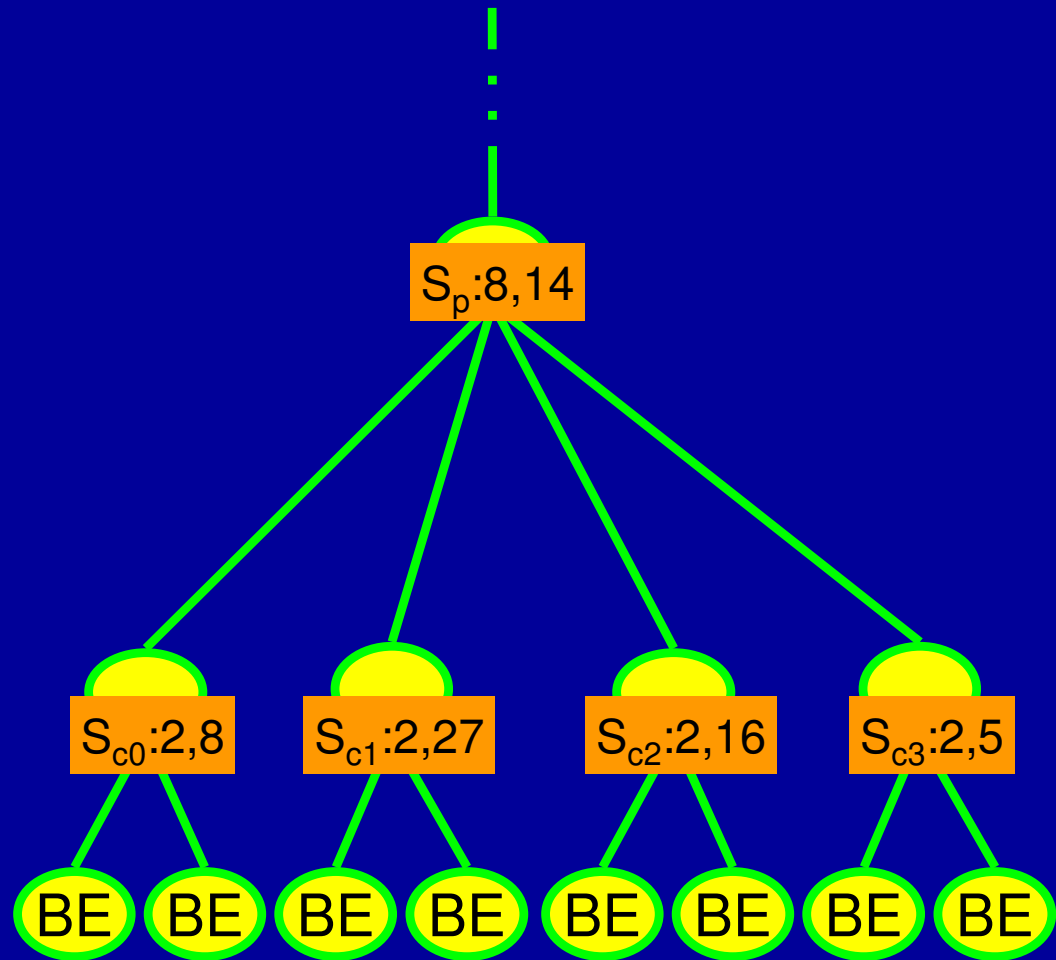
Reversibility Property of Stateful Reductions

- Reversible persistent state
 - There exists a **decomposition operation** that inputs the state of a process' parent and siblings and outputs the state of that process
- Partially reversible persistent state
 - There exists a decomposition operation that inputs the state of a process' parent and siblings and outputs state semantically equivalent to the state of that process

MMIC Example: Running Average Filter

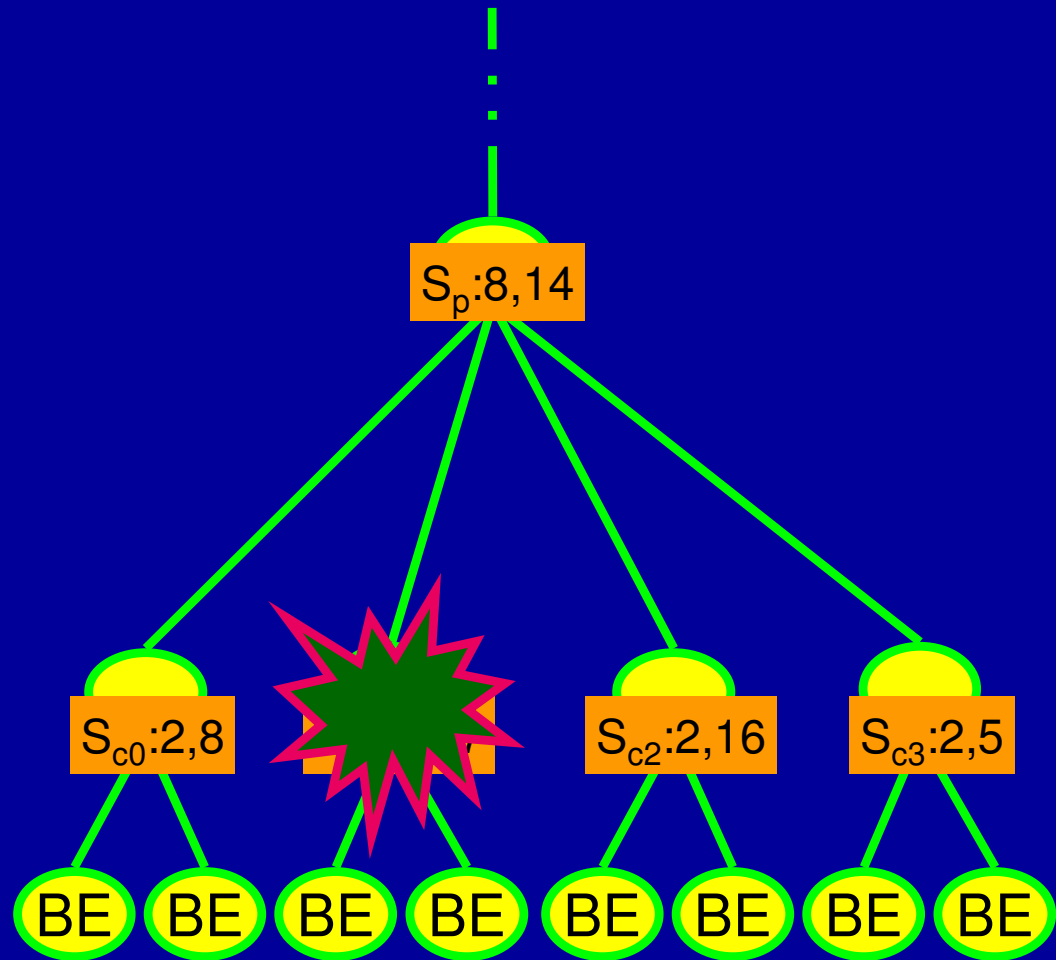


MMIC Example: Running Average Filter



MMIC Example: Running Average Filter

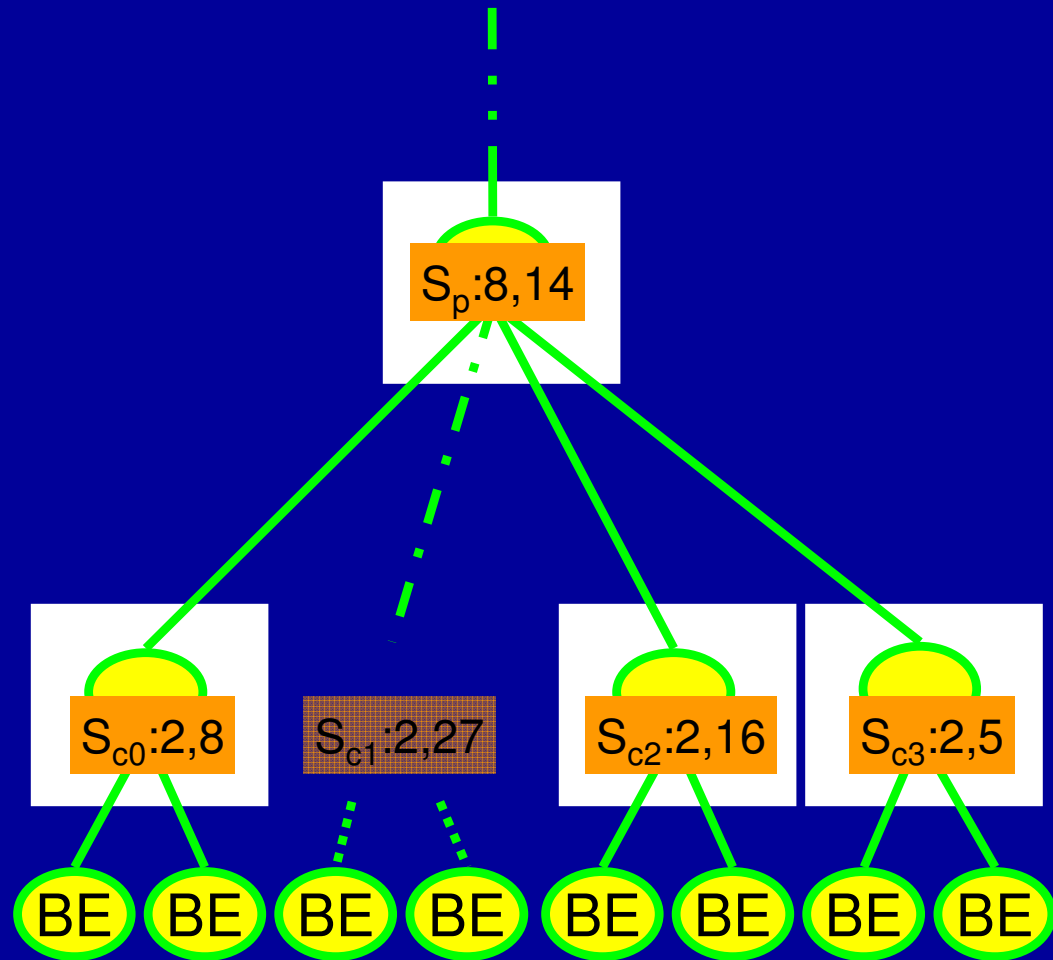
1. Detect Failure



MMIC Example: Running Average Filter

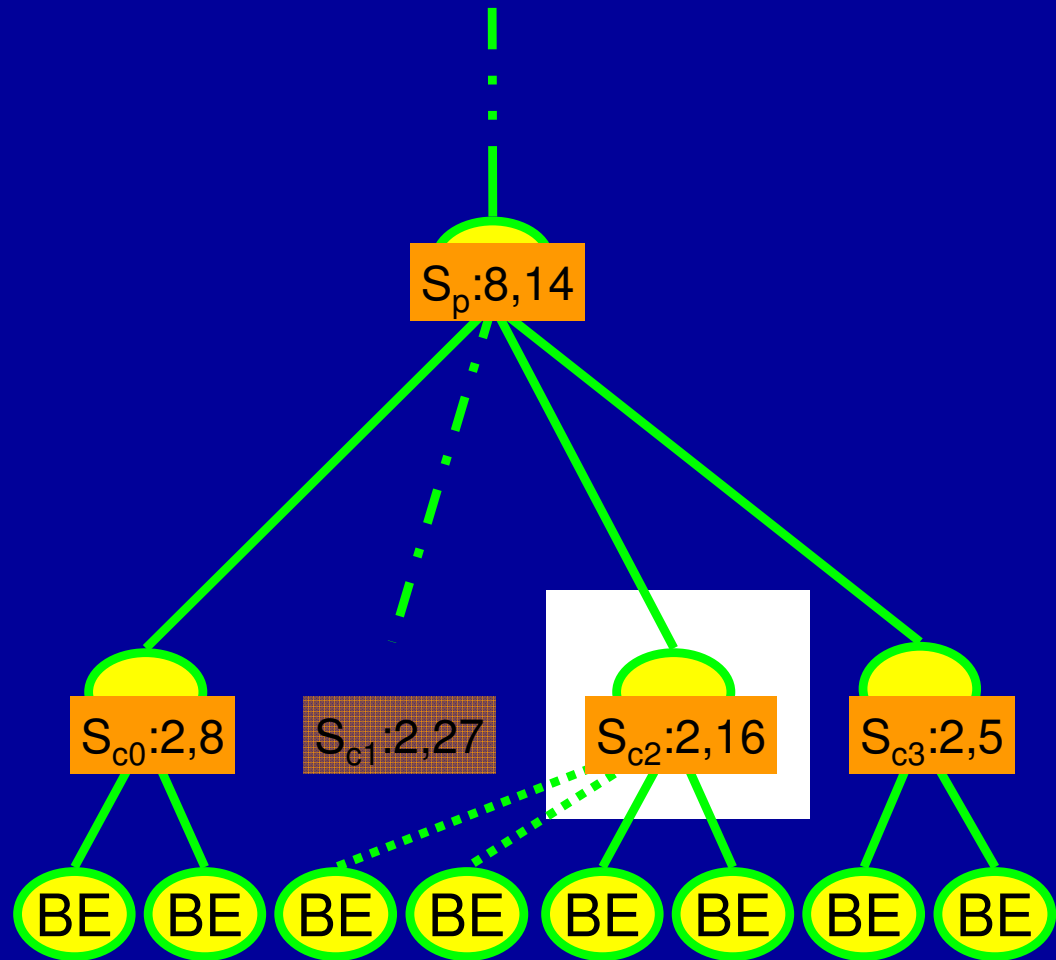
1. Detect Failure

2. Calculate Recovery Clique

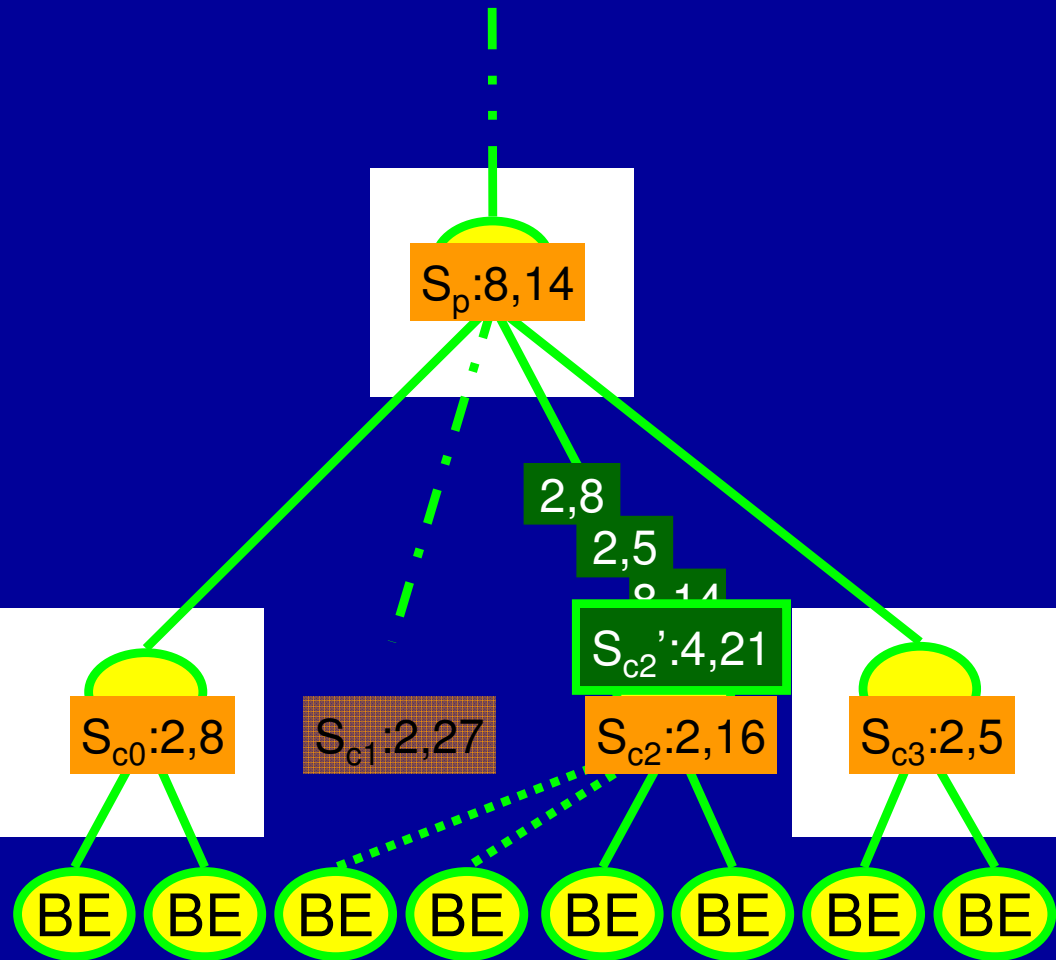


MMIC Example: Running Average Filter

1. Detect Failure
2. Calculate Recovery Clique
3. Assign a "take-over" node.



MMIC Example: Running Average Filter



1. Detect Failure

2. Calculate Recovery Clique

3. Assign a "take-over" node.

4. Regenerate lost state into "take-over" node:

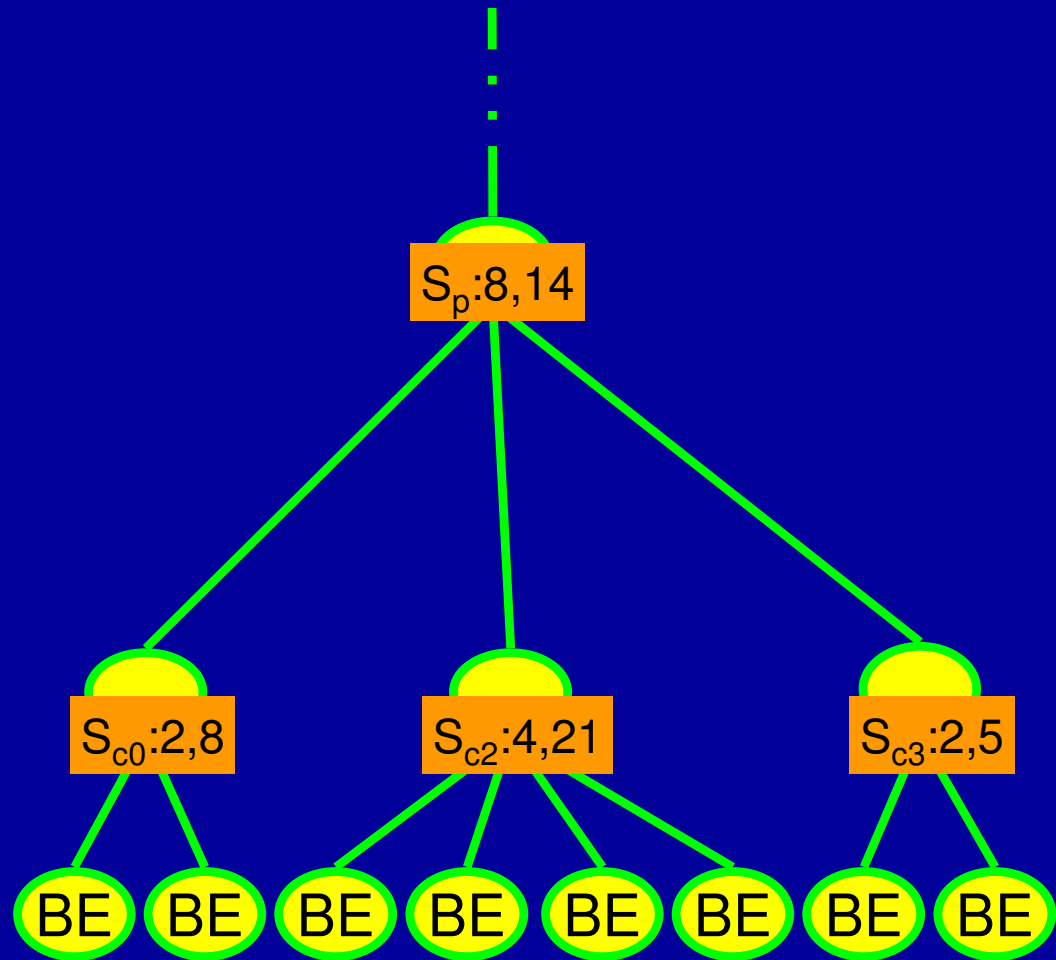
4.1 $\text{read}(S_p, S_{c0}, S_{c3})$

4.2 $\text{decompose}(S_p, S_{c0}, S_{c2}, S_{c3}) \rightarrow S_{c1}'$

4.3 $\text{merge}(S_{c1}', S_{c2}) \rightarrow S_{c2}'$

4.4 $\text{write}(S_{c2}') \rightarrow S_{c2}$

MMIC Example: Running Average Filter



1. Detect Failure

2. Calculate Recovery Clique

3. Assign a “take-over” node.

4. Regenerate lost state into “take-over” node:

4.1 $\text{read}(S_p, S_{c0}, S_{c3})$

4.2 $\text{decompose}(S_p, S_{c0}, S_{c2}, S_{c3}) \rightarrow S_{c1}'$

4.3 $\text{merge}(S_{c1}', S_{c2}) \rightarrow S_{c2}'$

4.4 $\text{write}(S_{c2}') \rightarrow S_{c2}$

5. Update and resume

Summary: Petascale Reliability

- TBÖNs are a promising paradigm for scalable and reliable tools for petascale systems
- MRNet provides tools with scalable performance
 - Efficient control, data collection, and data analyses
- We are developing scalable mechanisms for high degrees of reliability
 - Work in progress
- Think not what you can do for MRNet, but what MRNet can do for you!

Results/References

- Roth, Arnold, and Miller, "MRNet: A Software-Based Multicast/Reduction Network for Scalable Tools", in SC2003.
- Roth, Arnold, and Miller, "Benchmarking the MRNet Distributed Tool Infrastructure: Lessons Learned", in 2004 High-Performance Grid Computing Workshop.
- <http://www.paradyn.org/mrnet>