Clint Moulds
cmould1@gl.umbc.edu

“The MAFT Architecture for Distributed Fault Tolerance”

by Kieckhafer, Walter, Finn, and Thambidurai

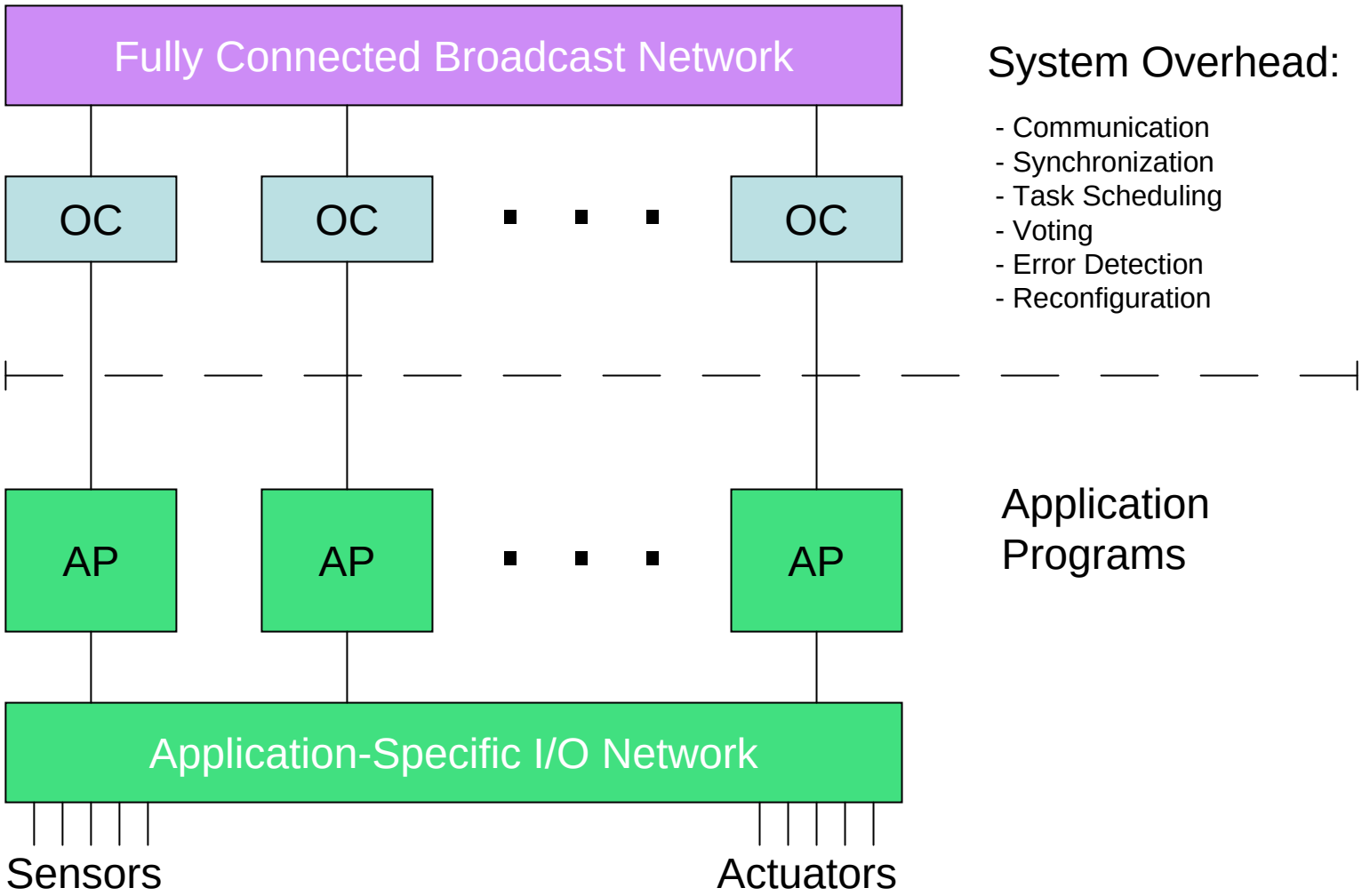
A summary presentation

Table of Section Headings

MAFT : Definitions & Objectives

- Multicomputer Architecture for Fault Tolerance
  - Distributed system for extremely reliable computation, without sacrificing performance, in real-time control systems. Design encompasses both hardware and software required.
- Ultrareliability required by intended applications
  - Control loop 200 Hz, 5.5 MIPS, I/O 1 MBPS, I->O delay 5ms
  - Probability of failure 10^(-10) per hour.
- As a result several “rare” faults must be handled
  - Multiple Coincident fault
  - Byzantine (malicious) fault
  - Generic (built-in) fault

MAFT Architecture



System Overhead:

- Communication
- Synchronization
- Task Scheduling
- Voting
- Error Detection
- Reconfiguration

Application Programs

MAFT Architecture

Partitioning of each node into 2 processors

- Operations Controller (OC)

  Handles executive functions to maintain the reliability of the system. Hardware-intensive. Connected to all other nodes' OCs via a broadcast network. (Size of MAFT system is strictly limited.)

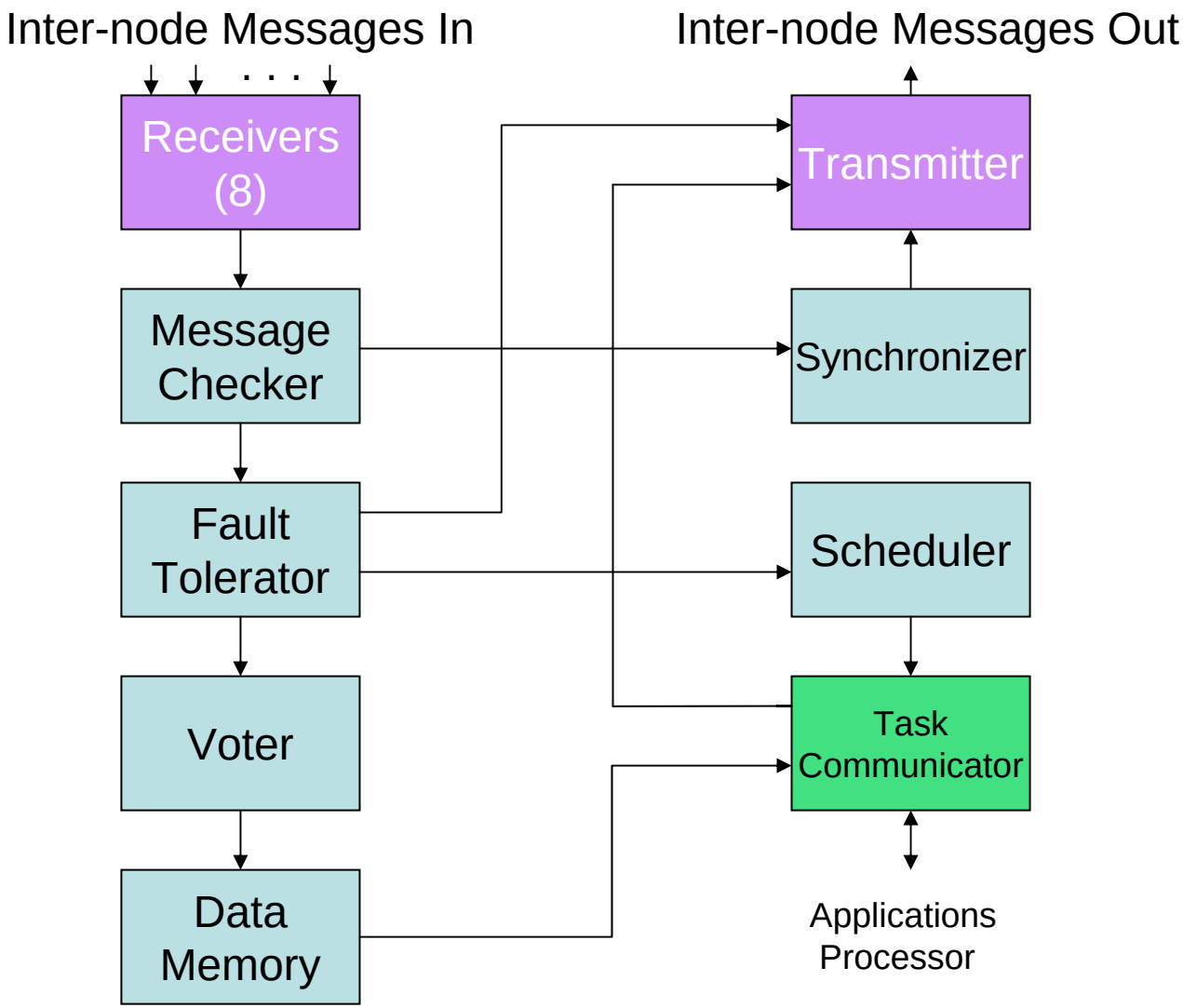
- Application Processor (AP)

  Runs typical operating system tailored to the given real-time application. Commonly performs control functions involving sensors and actuators.

Support for up to 8 nodes.

Allows for redundancy (some nodes running the same program) to handle multiple coincident faults. To handle generic faults, (like a bug in the PowerPC chip design, for example) MAFT allows each node to not only run different software but also different hardware. (This may be true only for the AP).

Operations Controller



OC : Communication Interfaces

- OC-OC
  - Dedicated serial broadcast link
  - 1 MBPS bandwidth per node
  - Dedicated receiver passes each message received to the Message Checker subsystem, which passes it along after several physical and logical checks.
- OC-AP
  - Asynchronous parallel “I/O device” in the Task Communicator subsystem. No need for special interface hardware or software in the AP.
  - AP completes a task, OC receives data value then gives AP the next task. OC’s may then vote on the value (it’s correctness) while the AP begins the next task.

OC : Communication Messages

- Data Message
  - Broadcast when a computed data value is received on the OC-AP.
- Scheduling Messages
  - [CS] : Completed/Started, broadcast when an AP completes a task.
  - [TIC] : Task Interactive Consistency, broadcast by all nodes at predefined intervals.
- Synchronization Message
  - [SS] : System State, broadcast by all nodes at predefined intervals.
- Error-Management Messages
  - [ERR] : Error Report, broadcast when any error is detected.
  - [BPC] : Base Penalty Count, provides a node just coming up with info on the error status of the entire system. Broadcast by all nodes at predefined intervals.

All Messages protected by link protocol and error control code (ECC), to guard against persistent corruption of messages.

OC : Synchronization Steady State

- Steady state synchronization is done on an “operating set” of working nodes, already synchronized with each other at least once.
- Uses a hardware version of Srikanth and Toueg’s optimal clock synchronization algorithm:
- “Presync” message
  - SS message, (which includes the local clock time).
  - The sender counts a predefined # of clock ticks after creating the SS message but before sending it. Each other node timestamps when they receive it.
  - Since these are sent out by all nodes at predefined intervals, the error estimate is the difference between the timestamp of a node’s own presync SS message and it’s “vote” (the most accurate value) on the entire set of timestamps.
- “Sync” message
  - A second SS message. The sender uses the error estimate calculated in the presync phase to adjust the # of clock ticks it counts before sending it out.

OC : Synchronization Startup

- Initial synchronization functions. “Cold start” is when all nodes are coming up, “Warm start” is when one node wants to join the operating set.

- Cold Start
  - Iteratively applies steady state algorithm of presync and sync messages for a pre-determined period of time. From these a synchronization interval is determined and voting yields a synchronization time.
  - Terminates through Byzantine Agreement. An ISW (“in sync with”) vector is included in each SS message. Consistency is ensured by comparing ISW’s in presync’s with those in sync’s. Once the consistent set is large enough an operating set is formed.
  - Large enough when # = max (3f+1, floor(N/2) + 1), where f = max number of maliciously faulty nodes.

- Warm Start
  - The node just starts broadcasting SS messages and converges to the voted value of the operating set’s timestamps.
  - Once synchronized (and hence evidenced by all) the nodes in the operating set may perform task reconfiguration to include the new node.

OC : Data Management Scheme

- Shared data values generated by applications are stored at every node. Provides N-way redundant storage and faster access of needed data.
- Data message sent out by a node is tagged by a data identification descriptor (DID) so other nodes know what it is.
- As copies of a given DID arrive at a node it performs on-the-fly voting to store the most correct value it can. No waiting for all copies to arrive, since one node may hang.
- Reasonableness and Deviance checks are performed on the data received. The source node is identified when a check fails. The deviance allowed is specified for each DID individually.
- Model allows for redundant computations on nodes running different software or even different hardware. On-the-fly voting allows for differing computation times, and deviance window allows for some randomness in algorithms. Using different software and hardware of course helps prevent generic errors.

OC : Data Management Voting Algorithms

- Two algorithms may be used for voting on application data. Which one chosen depends on which fault-tolerance properties best suit your application. Both are variants of the Fault-Tolerant Midpoint voting strategy.

- MS : “Median Select”

  - Selects the center value of an odd # of inputs, or averages the 2 center values for an even # of inputs. μ = floor ((N-1)/2)
  - More robust than MME since it discards more extremal values.
  - A malicious error can prevent it from converging, however.

- MME : “Mean of the Medial Extremes”

  - Discards the μ most extreme values and computes the mean of the two remaining extremes. μ = floor ((N-1)/3).
  - Convergent even when up to μ erroneous copies present, whether errors are malicious or not.
  - Guaranteed convergence rate of 1/2.

OC : Scheduling
Task Model

- Task - a block of code executed without interruption on one AP. Attributes include iteration frequency, relative priority, desired redundancy, and inter-task dependencies.

- Since MAFT considers all tasks as periodic, these inter-task dependencies (concurrent forks and joins, conditional branches) are a way to model nonperiodic behavior.

- Iteration period = 1 / frequency. Shortest period called the “atomic period”. SS messages sent out on atomic period boundaries (not necessary each one).

- Lowest frequency task defines the “master period”. Since all periods must be powers of 2, and MAFT allows up to 1024 atomic periods per master period, tasks may have up to 10 different frequencies.

OC : Scheduling Method

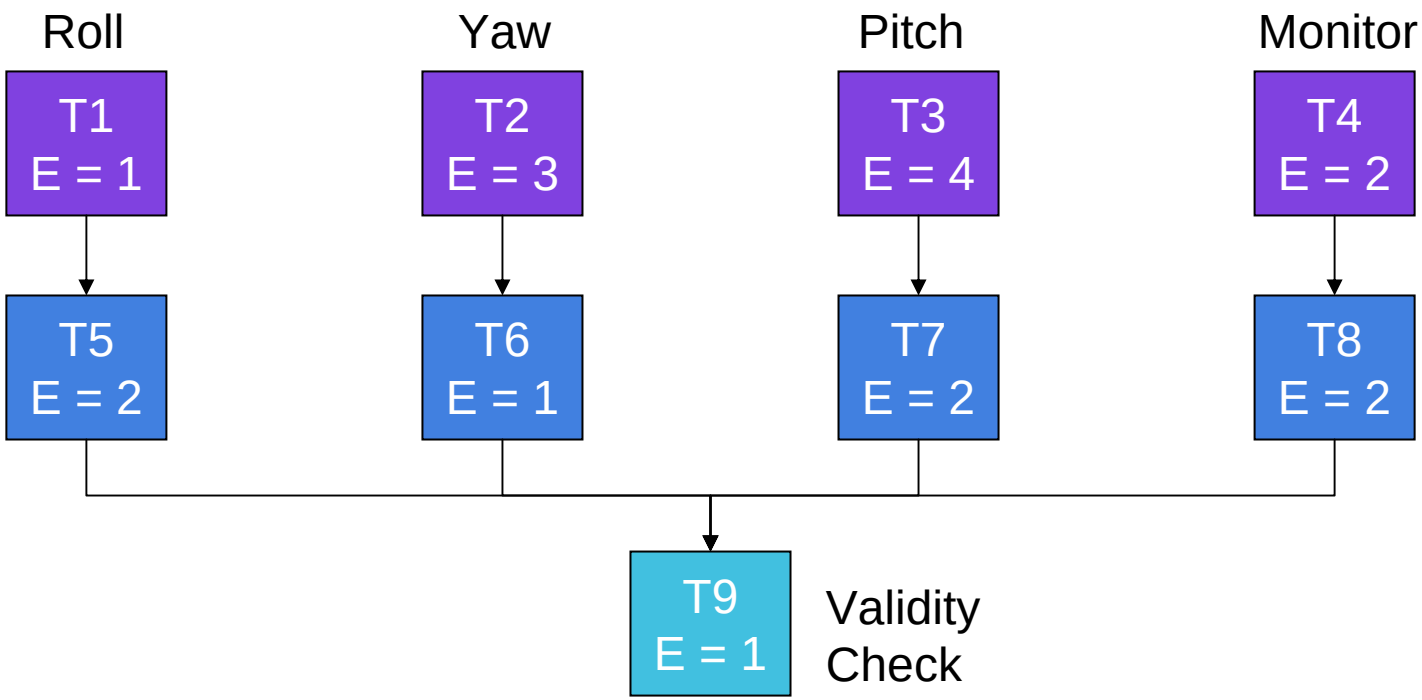
- Each node performs scheduling of tasks for its own node AND for every other node. The scheduling algorithm itself is a simple deterministic priority list, where the highest priority task ready for execution is assigned to an available node. Selections are monitored by all nodes in order to maintain Byzantine Agreement on scheduling data.

- Inter-task dependencies (forks, joins, conditional branches) are included in CS messages. Monitoring scheduling data is done using an Interactive Consistency algorithm. This will result in rebroadcasts when there are disagreements.

- TIC messages are used to synchronize these rebroadcasts. TIC's are sent out on "subatomic" period boundaries (another predefined value). A TIC has 2 bytes : a task completed byte indicating from which nodes a CS message was received, and a branch condition byte listing the inter-task dependencies in each of those CS messages. Hence Byzantine Agreement can be achieved.

- Task execution time as well as sequence of execution of tasks are also monitored. If a node detects a task is run out of sequence, or if the execution time is outside of a predetermined window (using execution timers synchronized to subatomic period boundaries), then an error is reported.

OC : Scheduling Example



Period	1	2	3	4	5	6	7	8	9	10	11	12
Group 1	*1*	*3*				*5*		*7*		ϕ		*9*
Group 2	*2*			*4*		*6*	ϕ	*8*		ϕ		*9*

OC : Task Reconfiguration Processes

- Reconfiguration algorithm involves tasks and nodes when the operating set changes. System enforces agreement on the members of the operating set before the algorithm is run. Output of the algorithm is an eligibility table, which indicates the nodes to which each task may be assigned.

- Three independent processes are used in running the algorithm:

  - Global Task Activation process : Based on the overall system capability, activates or deactivates tasks. (I.E. if a task cannot be supported in the current configuration, the GTA will deactivate it. Likewise it will activate an inactive task if the capability of the system later increases.)

  - Task Reallocation process : Maintains the desired redundancy of each task by reallocating tasks to nodes. Each task keeps a list of nodes in order of preference, and a level of redundancy ‘x’ is maintained using the ‘x’ most preferred nodes.

  - Task-Node Status Matching process : Matches tasks assigned to nodes with their operational status (included or excluded).

OC : Task Reconfiguration Terminology

- [N]
  Number of nodes in the operating set.

- [E_i]
  Task execution time in subatomic periods.

- [$R_i(N)$]
  Required redundancy of the task.

- [W(N)]
  Total processing time required for the workload. (Sum (Ei * Ri(N)) over i)

- [$t_0(N)$]
  Best-case total processing time for the workload. (W(N) / N)

- [t]
  Observed execution time for the workload.

- [AP Utilization]
  How close to optimal the AP usage (running tasks) is. ($t_0(N)$ / t)

OC : Task Reconfiguration Example

i	Ei	Ri(N)	Node Preference Order
1	1	3	1 2 3 4 5 6
2	3	3	4 5 6 1 3 2
3	4	3	1 2 3 4 6 5
4	2	3	4 5 6 2 1 3
5	2	3	1 2 3 5 4 6
6	1	3	4 5 6 2 3 1
7	2	3	1 2 3 6 5 4
8	2	3	4 5 6 3 2 1
9	1	N	1 2 3 4 5 6

Task-to-Node Preference Ordering

N	W(N)	t_0(N)	t	% AP Utilization
3	54	18.00	20	90
4	55	13.75	17-18	76-81
5	56	11.20	15-16	70-75
6	57	9.50	12	79

Multiprocessing Efficiency

(See previous slide for explanation of terms)

OC : Error Handling

- ERR message contains 31 possible error flags.
- The more errors a node accrues the higher its base penalty count (BPC) becomes. Incremental penalty count (IPC) is a weighted version of the BPC, reflecting the seriousness of the errors.
- All nodes track errors on all other nodes, and every atomic period ERR messages are broadcast and voted on. Once Byzantine Agreement is reached, any node whose BPC exceeds a certain threshold will be recommended for exclusion from the operating set. This recommendation is included in SS messages.
- Task reconfiguration would then take place either immediately or at the start of the next master period. The excluded node is masked completely until reconfiguration is complete, at which time it may resume some (limited) participation in the system. It may be assigned to run diagnostics, for example, until it's BPC falls to an acceptable level.
- Additional tests are run periodically to test the Error Detection mechanism itself, since the vast majority of the time there will not be errors. In these tests one node is chosen to broadcast a stream of erroneous bits. In these the IPC is added to the BPC to prevent exclusion of this node.

Conclusion

- MAFT has been implemented in two prototypes so far and has been shown to meet it's performance goals. Control loop frequencies of 350 Hz, I/O bandwidth of 1 MBPS, throughput 5.5 MIPS. It maintains Byzantine Agreement on system parameters using interactive consistency and convergent voting algorithms. On-the-fly voting and deviance checking support mixing and matching hardware and software in the system.