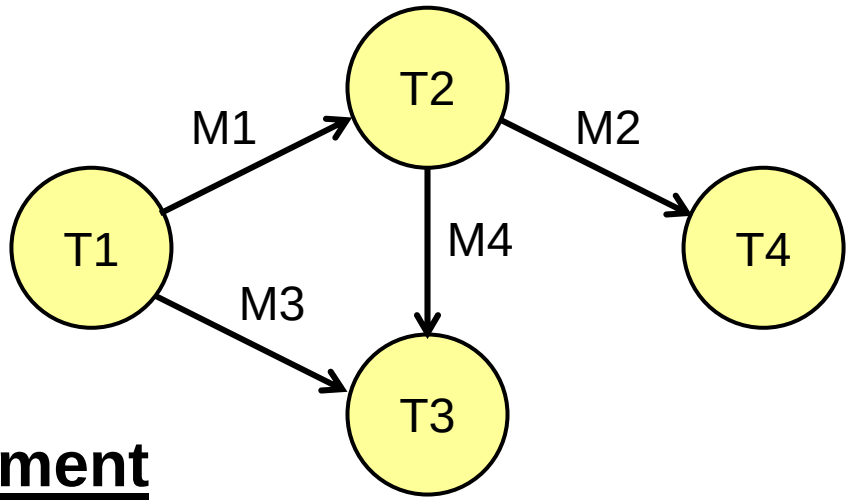
Distributed System Exploration

Mirabilis Design Inc.

Design Analysis 2: Distributed System

- System details
  - Multi-independent processing computers
  - Software tasks distributed across these computers
  - Connectivity across multiple shared networks
- Analysis
  - Optimal Routing Table configuration
  - Capacity planning
  - Software tasks and thread distribution
  - Resource allocation

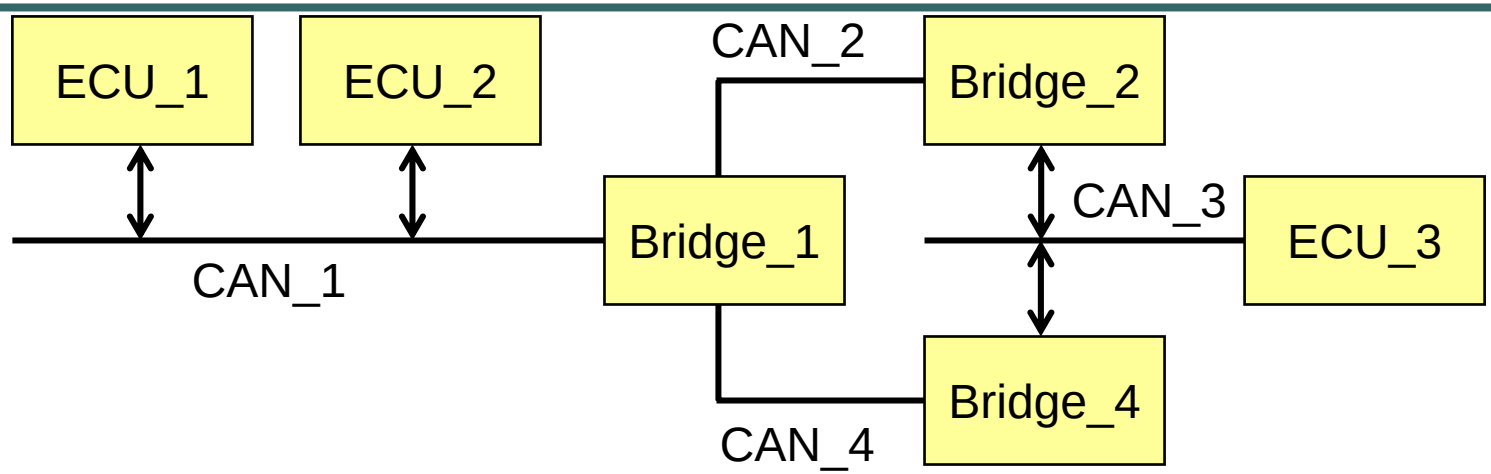
Example 2: Distributed System Logical Task Flow



Initial Assignment

List of Tasks: T1, T2, T3, T4
T1, T2 -> ECU_1
T3 -> ECU_2
T4 -> ECU_3
Messages: M1, M2, M3, M4

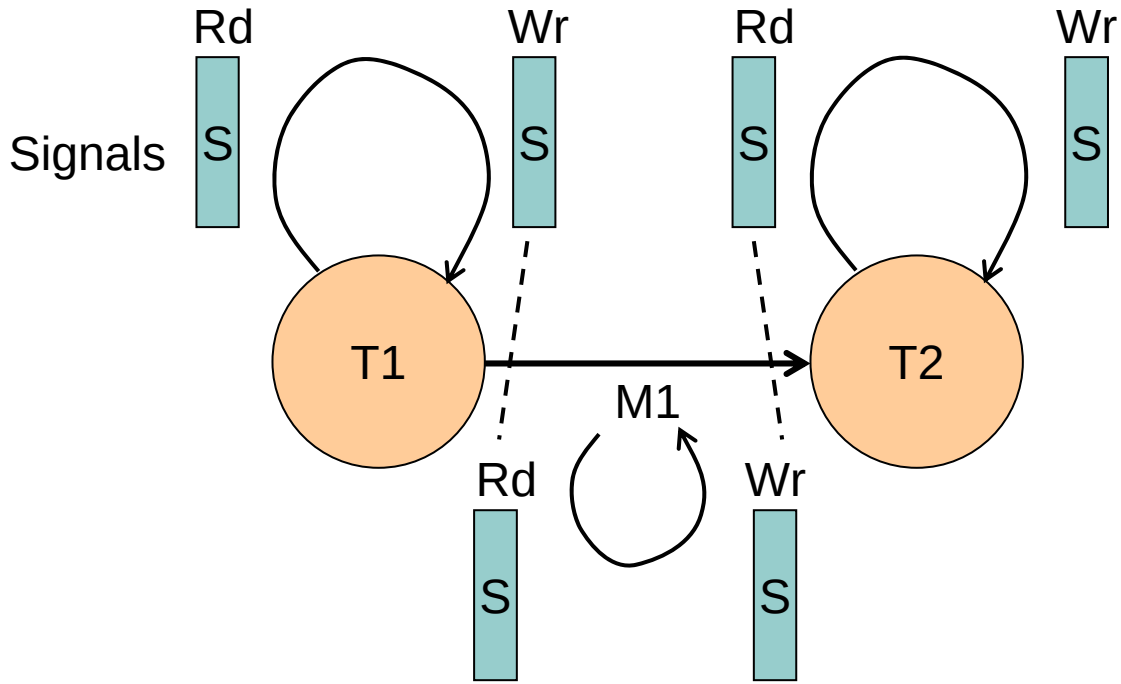
Example 2: Distributed System uP-Computer/Bus Physical Mapping



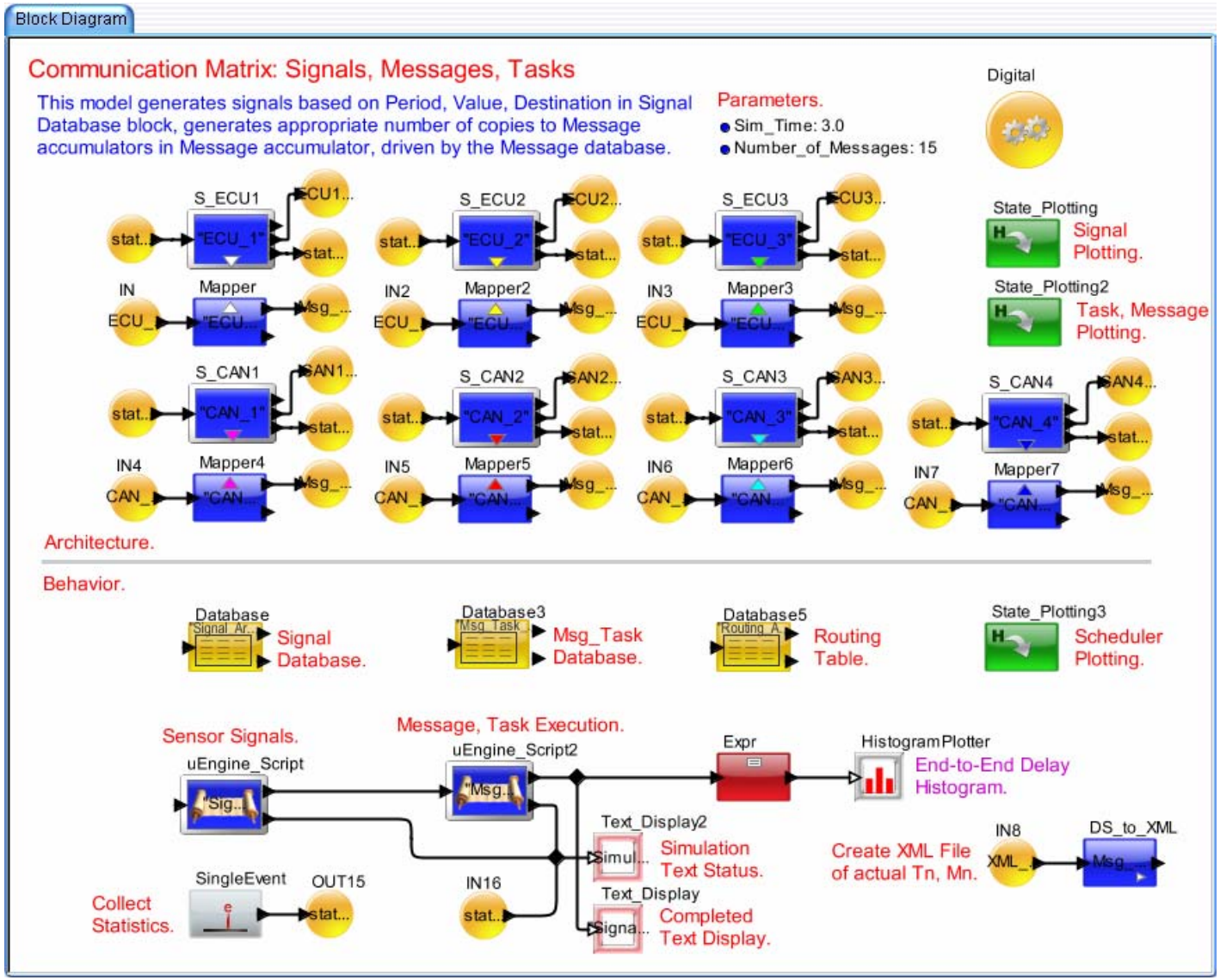
Message Assignment

- Messages: M1, M2, M3, M4
- Internal to ECU, if Source ECU == Destination ECU
- CAN_N bus segments selection based on Source:Destination of Task,
- Routing is selected for shortest hops
- Dynamic allocation based on topology selection
- An ECU is the equivalent of a core, processor or computer.

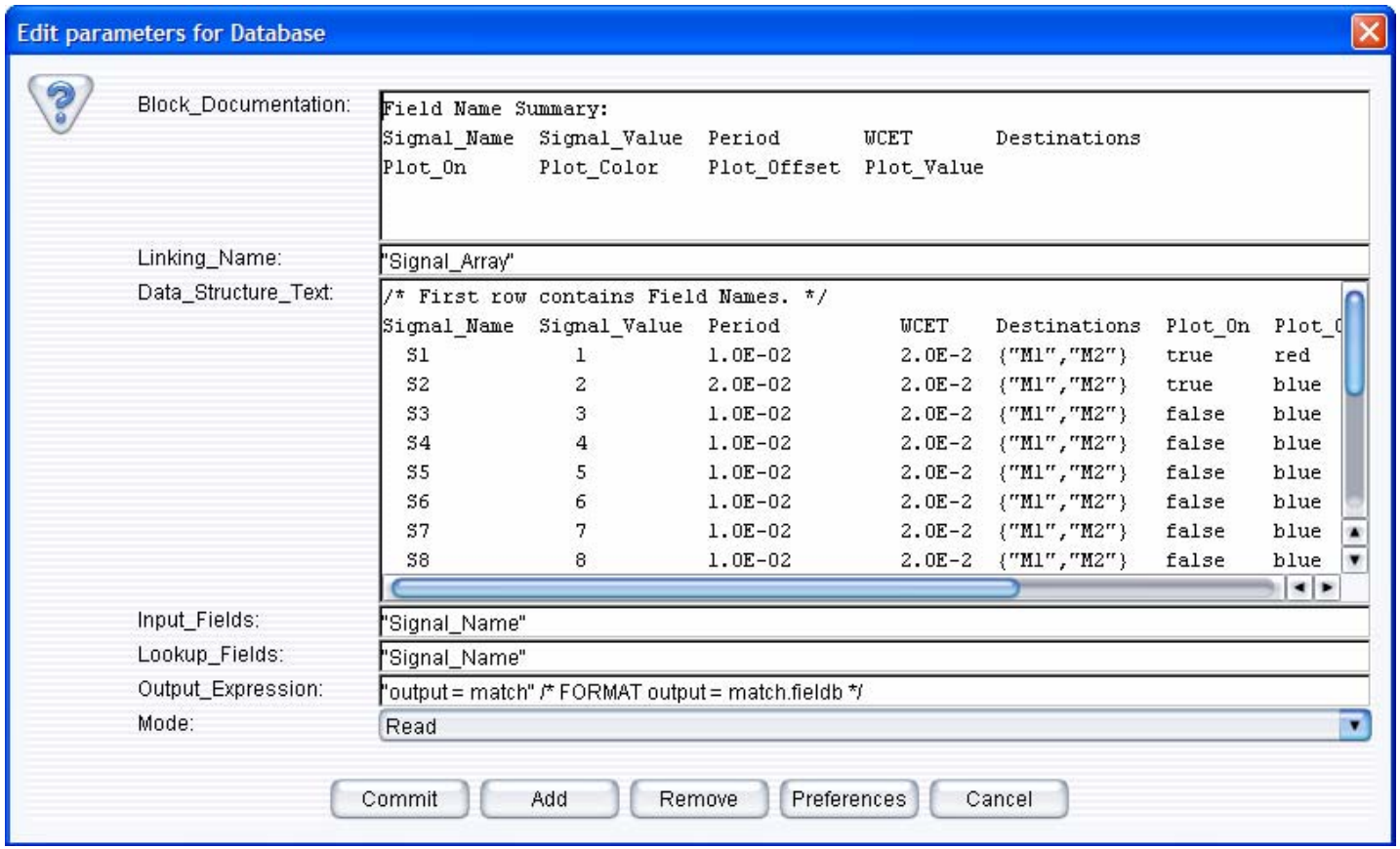
End-to-End Latency



- T1 writes latest S1(time) when task executes.
- M1 reads T1(time) via Middleware Buffer when it periodically executes, sends packet through Bus structure.
- T2 reads M1(time) from buffer when it fires via Middleware Buffer, and if the last node, outputs end-to-end latency.

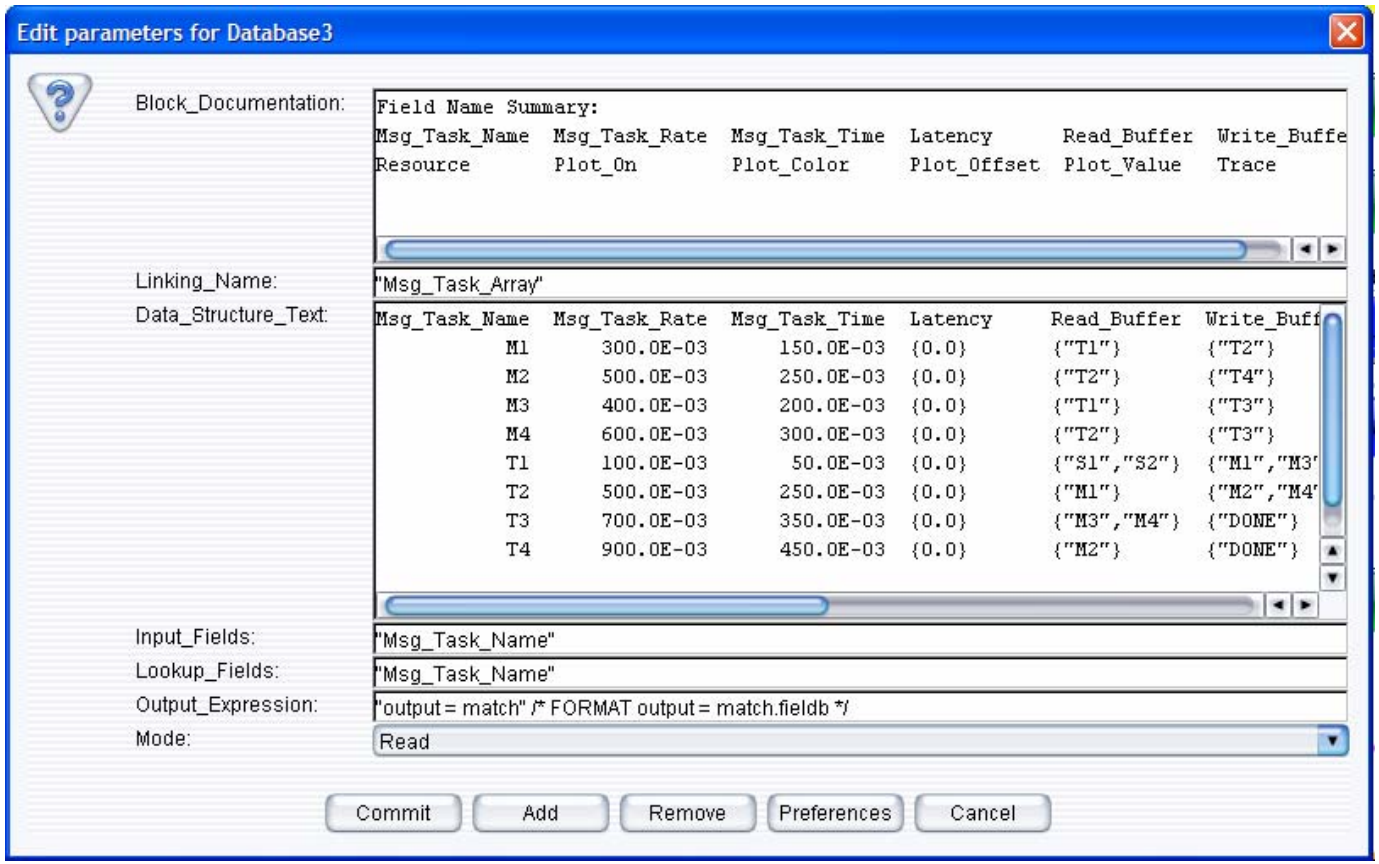

Block Diagram
Communication Matrix: Signals, Messages, Tasks
This model generates signals based on Period, Value, Destination in Signal Database block, generates appropriate number of copies to Message accumulators in Message accumulator, driven by the Message database.
Parameters.
Sim_Time: 3.0
Number_of_Messages: 15
Digital
State_Plotting
Signal Plotting.
State_Plotting2
Task, Message Plotting.
S_ECU1
S_ECU2
S_ECU3
Mapper
Mapper2
Mapper3
S_CAN1
S_CAN2
S_CAN3
S_CAN4
Mapper4
Mapper5
Mapper6
Mapper7
Architecture.
Behavior.
Database
Signal Database.
Database3
Msg_Task Database.
Database5
Routing Table.
State_Plotting3
Scheduler Plotting.
Sensor Signals.
Message, Task Execution.
uEngine_Script
uEngine_Script2
Expr
HistogramPlotter
End-to-End Delay Histogram.
Text_Display2
Simulation Text Status.
Create XML File of actual Tn, Mn.
DS_to_XML
SingleEvent
OUT15
Collect Statistics.
IN16
Text_Display
Completed Text Display.

Signal Table



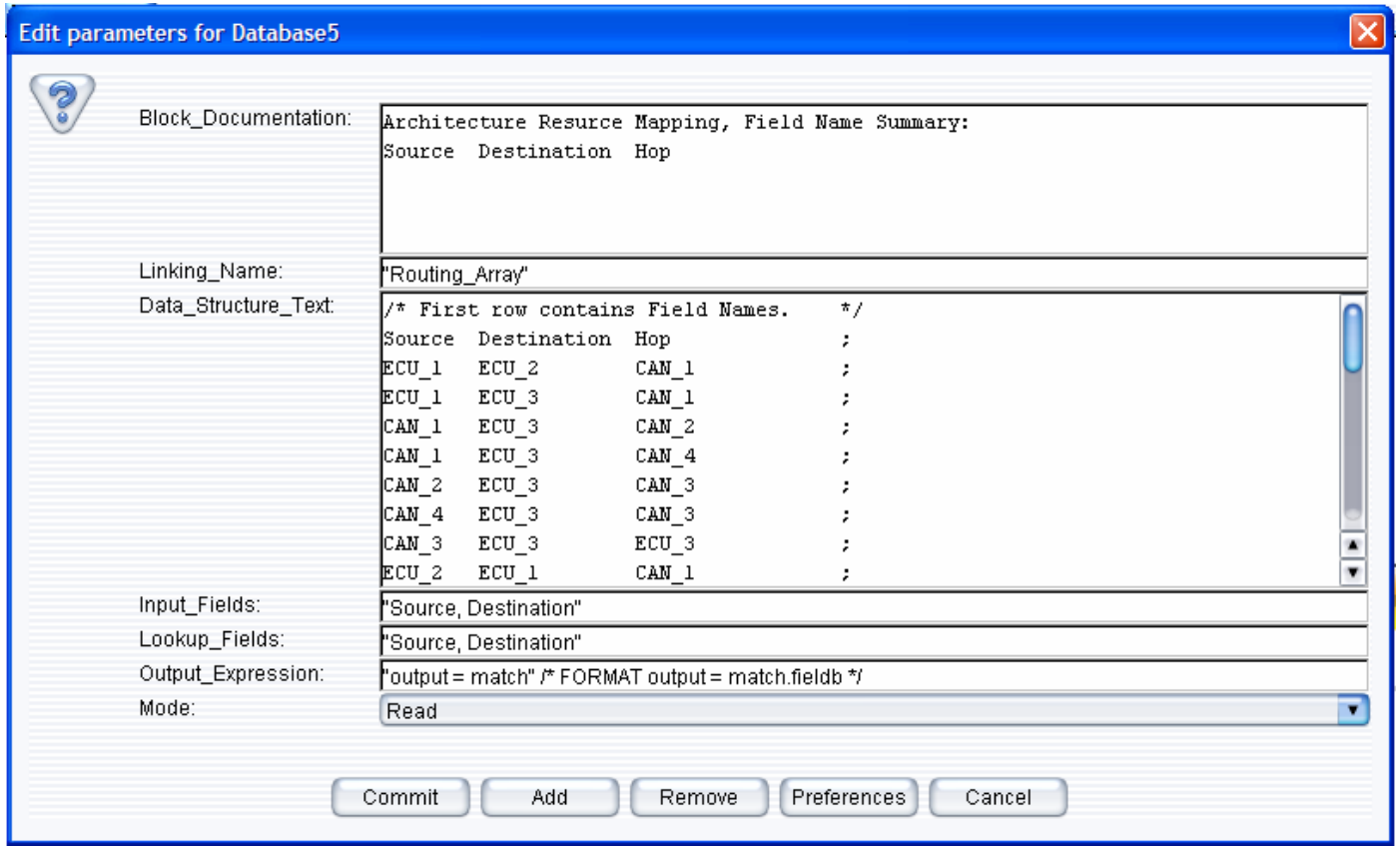
Note: Destinations can be Tasks, or Messages

Task/Message Table



Note: Read_Buffer, Write_Buffer columns can designate Tasks or Messages

Routing Table



Note: Simple Source, Destination, Hop format

Discovery Process

Discover Logical Path: M1 Source: T1 Destination: T2
Discover Physical Path: M1 Source: ECU_1 Destination: ECU_1

First Hop Array:
{"NONE"}

No Further Processing Required...

Discover Logical Path: M2 Source: T2 Destination: T4
Discover Physical Path: M2 Source: ECU_1 Destination: ECU_3

First Hop Array:
{"CAN_1"}

Second Hop Array:
{"CAN_2", "CAN_4"}

Third Hop Array:
{"CAN_3"}

Fourth Hop Array:
{"NONE"}

Final Hop Array:
{{"CAN_1", "CAN_2", "CAN_3"}
, {"CAN_1", "CAN_4", "CAN_3"}
}

Discover Logical Path: M3 Source: T1 Destination: T3
Discover Physical Path: M3 Source: ECU_1 Destination: ECU_2

First Hop Array:
{"CAN_1"}

No Further Processing Required...

Final Hop Array:
{{"CAN_1"}
}

Discover Logical Path: M4 Source: T2 Destination: T3
Discover Physical Path: M4 Source: ECU_1 Destination: ECU_2

First Hop Array:
{"CAN_1"}

No Further Processing Required...

Final Hop Array:
{{"CAN_1"}
}

End-to-End Delay

- End-to-End Delay is composed of the following elements:
  - Task – Signal or Message (Signal) Middleware Buffer, Periodic Task time, ECU Execution time
  - Message – Task Middleware Buffer, Periodic Message time, Bus(n) Transfer time
- ECU Execution time:
  - Time due to arbitration/queueing, due to processing
- Bus Transfer time:
  - Time due to arbitration/queueing, due to transfer for each Bus segment in a path.

Note: Receive Interrupt, or Polling can be added

End-to-End Delay Equation

$$Latency = Task(i)_{Sample} + Task(i)_{Exec} + Msg(i+1)_{Sample} + Msg(i+1)_{Xmit}$$

where

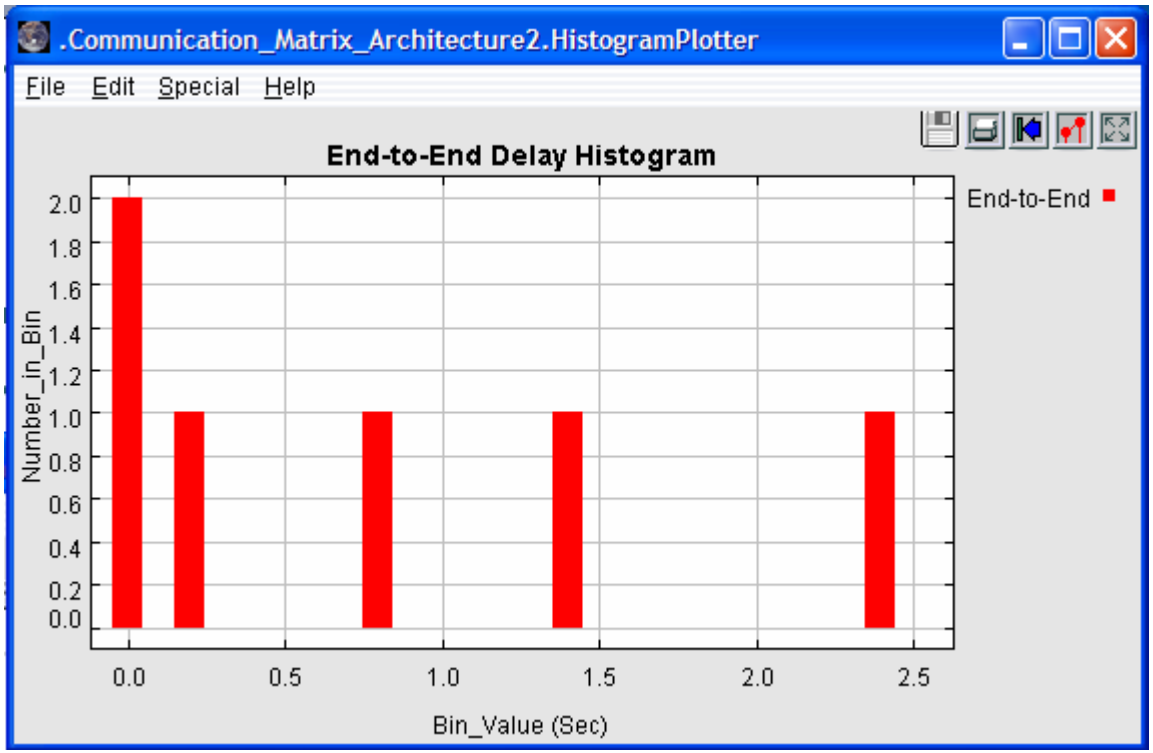
$$Task(i)_{Sample} = Sampling\_Delay\_of\_Application\_Software$$

$$Task(i)_{Exec} = Task\_Exec\_Time + Blocking\_due\_to\_Higher\_Priority$$

$$Msg(i+1)_{Sample} = Sampling\_Delay\_of\_Message$$

$$Msg(i+1)_{Xmit} = \Sigma Message\_Xmit\_Time + Blocking\_due\_to\_Higher\_Priority$$

End-to-End Delay Histogram



Task, Message Execution

Model Assumptions

- Each Task will execute periodically for a specified time on an ECU, read from Middleware Buffers, or write a Signals from a prior Task from Middleware Buffers, desinated in a Table.
  - Columns: Read_Buffer   Write_Buffer
- Each Message will execute periodically for a specified time, read from Middleware Buffers and send via Buses to a distant Middleware Buffer.
  - Same Columns: Read_Buffer   Write_Buffer
- If a Task accepts two Middleware Buffers, each will be retained in an array to maintain proper end-to-end latency, or follow-on processing.

Model Assumptions (cont)

- Deployment of Tasks to ECUs and Messages to Buses
  - Discovery process is flexible enough to allow the specification of the mapping of tasks (t1, t2, t3, t4) to ECUs and the automatic determination of the mapping of messages (m1, m2, m3, m4) to buses.
  - User can map Tasks to ECUs in a table format.
- After Task completes, it will pass the transmit message, or receive message, through bus segments to continue the process, unless destination is “NONE”.
- Model will select M1, M2, M3… paths during initialization of model and insert the shortest path into database memory for each Task.

Model Advantages

- A Table-Driven model should scale simply by modifying the table entries.
- A Table-Driven model should allow entry of physical mapping, or routing, from existing sources, some pre-processing to get into source, destination, hop format.
- User can allocate Signals, Messages, Tasks, to perform What-If scenarios.
  - A single Script-based block can support 32 independent periodic signals, tasks, or messages currently, and Mirabilis can provide 64 or 128 per block.
  - One Table might be created and allocated to different tables in a model.
- Additional Task logic could be added, either in script form, block style, or C code.
- Data structure fields can capture the internal flow with variable length arrays containing physical names in one array and timestamps in a double array.