Whether communication is by messages or by channels, MCAPI requires a sending node to have available an endpoint on the receiving node in order to communicate. Thus there is the following discovery and bootstrapping issue in MCAPI when static naming is not in use (and in any dynamic communication API for that matter): How to transfer the first endpoint from a receiver to a sender? This section proposes a discovery model, which addresses the issue of how an MCAPI sender can bootstrap itself by finding the endpoint for a receiver, essentially before any user-level communication has been established.

The basic idea is to allow the MCAPI runtime system to facilitate the creation of a root endpoint that is visible to all the nodes in the system. An endpoint can "publish" itself to the communications layer as the "root" or first-level name server of the namespace using statically known domain and node numbers. For example, the system could have exactly one statically numbered node, domain 0, node 0. All other nodes in the system would then register with node 0 via asynchronous messaging, sending it messages to let it know that they exist and what endpoints they have dynamically created. Similarly, dynamic nodes and endpoints can discover each other by sending messages to the 'root node'. The format of the message is user defined.

7.3 Automotive Use Case

7.3.1 Characteristics

7.3.1.1 Sensors

Tens to hundreds of sensor inputs read on periodic basis. Each sensor is read and its data is processed by a scheduled task.

7.3.1.2 Control Task

A control task takes sensor values and computes values to apply to various actuators in the engine.

7.3.1.3 Lost Data

Lost data is not desirable, but old data quickly becomes irrelevant, most recent sample is most important.

7.3.1.4 Types of Tasks

Consists of both control and signal processing, especially FFT.

7.3.1.5 Load Balance

The load balance changes as engine speed increases. The frequency at which the control task must be run is determined by the RPM of the engine.

7.3.1.6 Message Size and Frequency

Messages are expected to be small and message frequency is high.

7.3.1.7 Synchronization

Synchronization between control and data tasks should be minimal to avoid negative impacts on latency of the control task; if shared memory is used it will always be one task writing and the other reading, so synch is not required or desirable; deadlock will not occur but old data may be used if an update is not ready.

7.3.2 Key functionality requirements

7.3.2.1 Control Task

There must be a control task collecting all data and calculating updates; this task must update engine parameters continuously; Updates to engine parameters must occur when the engine crankshaft is at a particular angle, so the faster the engine is running, the more frequently this task must run.