

MP-SoC June 2007

*Formal modeling and a network centric  
Real-Time Operating System  
in less than 2K Bytes as a generic base  
for MP-SoC and Process Oriented  
Programming*

[www.OpenLicenseSociety.org](http://www.OpenLicenseSociety.org)  
[www.melexis.com](http://www.melexis.com)

Unifying and systematic system development methodologies  
with trustworthy embedded components

[Eric.Verhulst@OpenLicenseSociety.org](mailto:Eric.Verhulst@OpenLicenseSociety.org)

28/06/2007

Open License Society

1



## Who is Open License Society?

- Privately funded R&D institute
  - Leuven (BE), Berdyansk (UA)
- Why: 70 % of all SE projects do not deliver
- Objectives
  - Systematic & Unified Systems Engineering Methodology
  - 'Interacting Entities' paradigm at all levels:
    - OpenComRTOS as runtime environment (formal developed)
  - Implies '**Trustworthy Components**'
    - => **Open License** (source code + all design, test, .... docs)
- Focus:
  - Embedded Systems:
    - Constraints driven development
    - Real-time, distributed, hardware & software, ...

28/06/2007

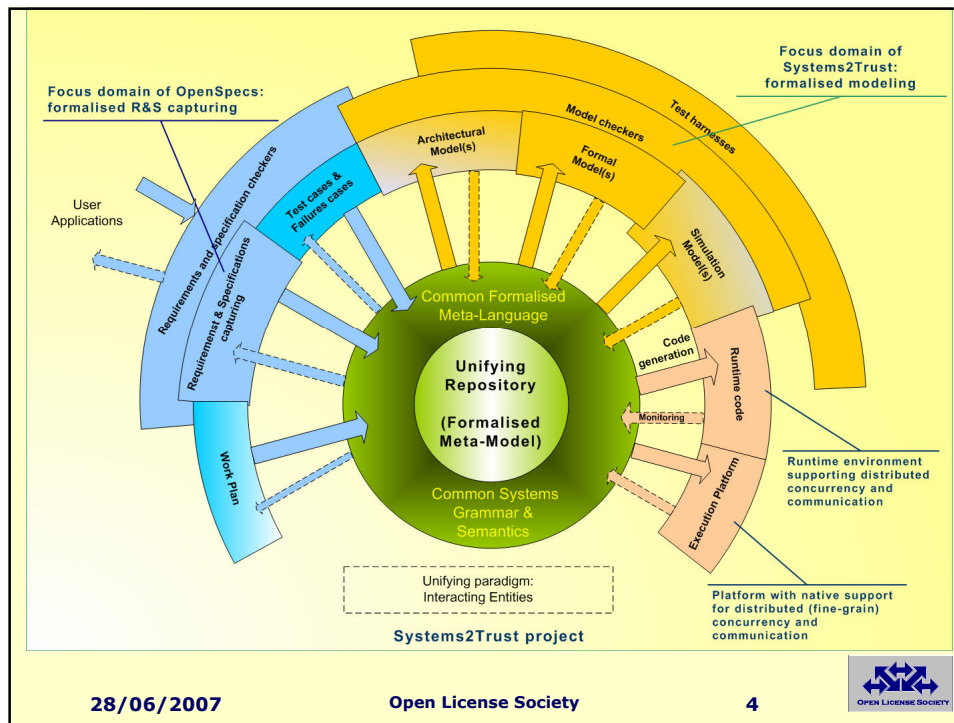
Open License Society

2

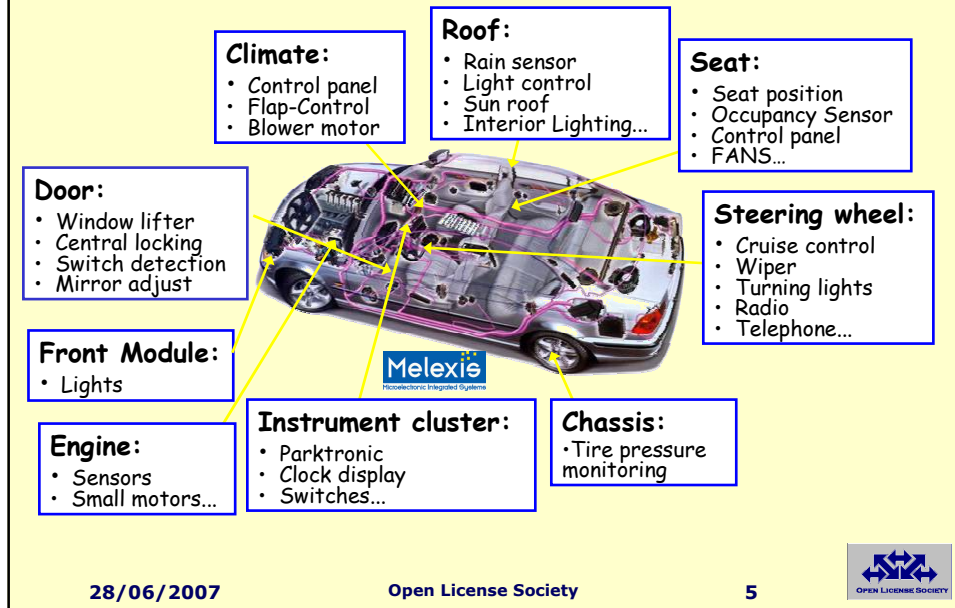


# Some keywords

- **Unified semantics:**
  - For all "views" (from requirements to platform)
  - Full behavior (at all levels)
  - Interface definition (is more than syntax)
  - "protocols" rather than messages
- **System's grammar**
  - Defining a formalised language
- **Meta-modeling**
  - Raising the level of abstraction first
- **Interacting Entities**
  - Then define the architectural level=
    - Entities and Interactions
    - Applies to almost any system domain



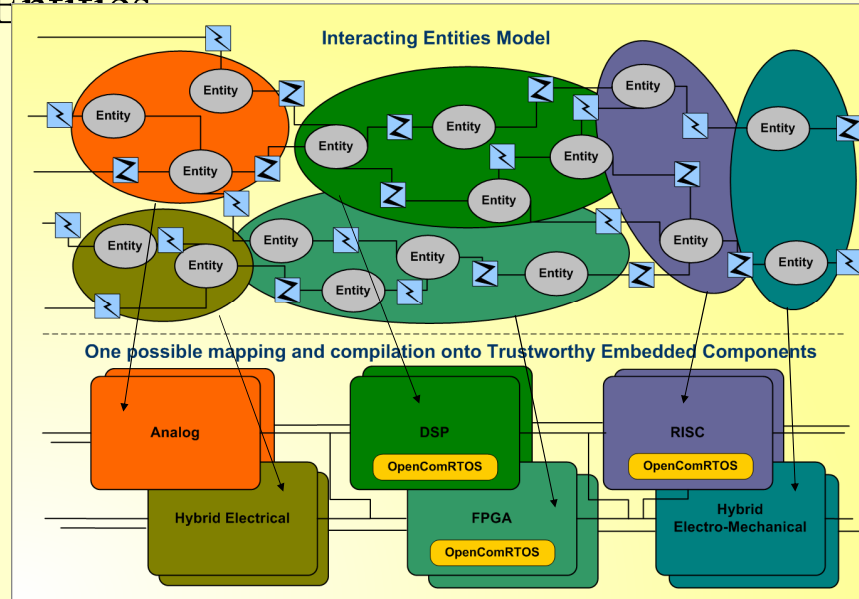
## Embedded Systems: safety first



## Runtime environment (software)

- Entities and their interactions are 'linked' with runtime components
- Ideally = proven and tested (=validated)
- Extra boundary conditions:
  - Real-time behaviour, performance, power consumption
  - Cost and size
  - Should be correct by design
  - Should be scalable by design
  - Should be safe and secure by design
  - Should support graceful degradation
  - Monitoring for confidence and post-fault analysis

# Unifying paradigm (1): Interacting Entities



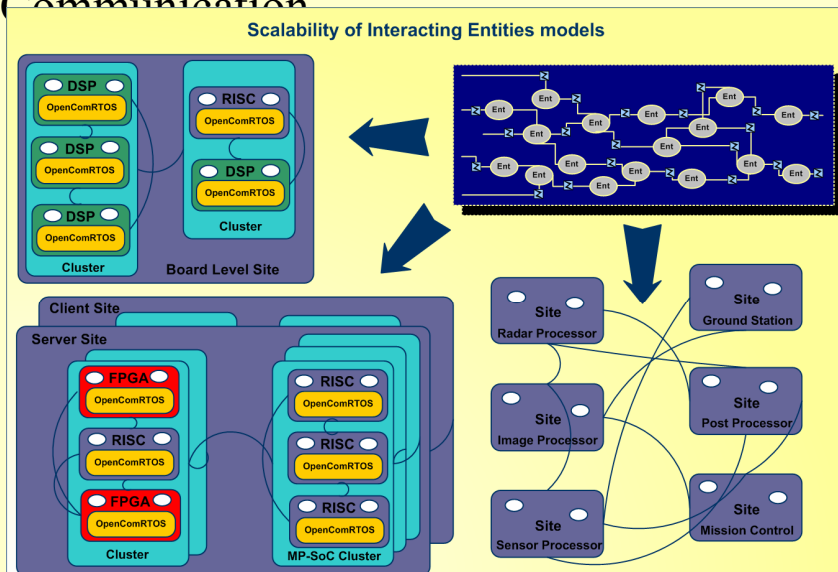
28/06/2007

Open License Society

7

OPEN LICENSE SOCIETY

# Unifying paradigm (2): Scalable Communication



28/06/2007

Open License Society

8

OPEN LICENSE SOCIETY

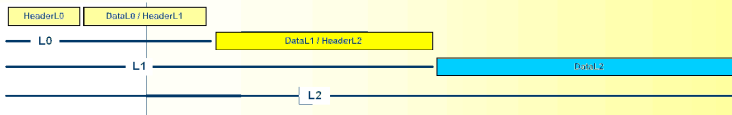
# OpenComRTOS: formally developed

- variable size packets
- widely distributed addressing
- dynamic protocol packets
- extensible API

- fixed size packets
- cluster addressing
- dynamic protocol packets
- API emulation

- fixed size packets
- tightly clustered addressing
- static protocol packets
- system packets
- scheduler
- routing and buffering
- runtime monitor

### Packet structure



These are semantic levels first of all:  
What behavior to expect?

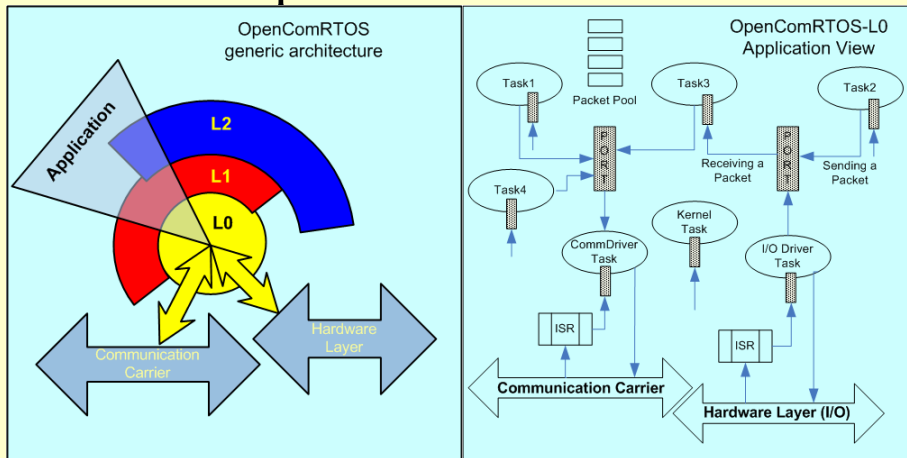
28/06/2007

Open License Society

9



# Generic Open-Comm-RTOS



Based on (scalable) "packet switching" at all levels

Tasks (entities) and interactions decoupled

28/06/2007

Open License Society

10



## Some requirements

- **Targets:**
  - Single chip, tightly coupled: multi-core
  - Multi-chip, tightly coupled: parallel processors on board
  - Multi-boards, multi-rack: using backplane interconnects
  - Distributed: using LAN and WAN
  - Host node (e.g. to use host-OS services and legacy)
- **Application: mix of distributed control and dataflow**
- **Programming models:**
  - "Interacting Entities"
  - "Virtual Single Processor":
    - transparent for topology
    - Supporting heterogenous targets
  - Distributed real-time (preemptive, priority based, timer based)
  - Safe, secure
  - Small code size, low latency (=high performance)

28/06/2007

Open License Society

11



## Formal modeling for developing OpenComRTOS

- **Goal:**
  - Develop Trustworthy distributed RTOS
    - Follow OLS SE methodology
    - Formal verification & analysis: formal modelling
  - Scalable distributed RTOS
  - Verify benefits and issues of using Formal Modeling
- **Why do we need formal techniques?**
  - How precise is the engineer's brain?
  - How precise is the management's brain?
  - How precise can we define requirements?
  - How precise can we define specifications?
  - How precise can we « write » software?
  - How precisely do we know all dependencies?
  - How sure can we be of the end-result?

28/06/2007

Open License Society

12



## Can we trust our mind ?

- How many « F » did you find ?

**FINISHED FILES ARE THE RESULT OF YEARS OF SCIENTIFIC STUDY COMBINED WITH THE EXPERIENCE OF YEARS**

Did you see the similarity with source code (debugging) ?

28/06/2007

Open License Society

13



## Formal modeling tools

- Default mathematical approach:
  - Correctness by proof
    - Labor and time intensive
    - Needs specialists
    - (Human) Error prone process
  - Tools needed
    - State space is exponentially large
    - Issues always in « hidden corners »
    - Allow incremental process
  - Requirements:
    - Support state machines
    - Support concurrency and communication
    - Low notational barrier

28/06/2007

Open License Society

14



## Formal modeling tools: selected options

- Investigated:
  - SPIN, B, CSP/FDR, **TLA+/TLC**
- Outcome of process:
  - SPIN OK, initially preferred, good documentation, wide user base, but very C-like style
  - CSP: hard notation, FDR not readily available
  - B: waiting for Event B, incremental approach and compositionality very good
  - TLA+/TLC
    - Based on Temporal Logic
    - Mathematical notation, but standard
    - Works for any domain (SW, HW, ...)
    - (but not for large models)

28/06/2007

Open License Society

15



## Benefits of TLA+/TLC

- TLA+/TLC home page on <http://research.microsoft.com/users/lamport/tla/tla.html>
- Initial models reflected "programming style"
  - That's the way the mind works (after being conditioned ...)
  - > 28 successive models from 2 pages to 25 pages
    - Initially very abstract, neglecting details
    - All successive models were correct, why ?
      - Iterative, incremental process!
      - Takes 15 minutes from one model to the next
    - Interplay between software architects and formal modeling engineer
      - Architectural model polluted by programming concepts
      - Abstraction from TLA helped to find these issues
      - Result: much cleaner, safer and performant architecture
- TLA models do not prove software is correct (! ?)
  - TLC proves that Formal **Models** are correct

28/06/2007

Open License Society

16



# Formally modeled

```

TypeInvariant == /\ ppool \in [Adr-> Packet \union {NoData}]
                /\ PQ \in [ FIFO : [Port -> Seq(Adr)],
                            WL  : [Port -> Seq(Adr)]]
                /\ chan \in [ val: [HLink -> Packet \union {NoData}],
                            stt: [HLink -> {"free","busy"}]]
                /\ TxQ \in [TxChan -> Seq(Packet)]
                \*   /\ tstate \in [UTask -> {"running","ready","wait4anS","wait4anR"}]
    
```

```

67 TypeInvariant  $\triangleq$   $\wedge$  ppool  $\in$  [Adr  $\rightarrow$  Packet  $\cup$  {NoData}]
69            $\wedge$  PQ  $\in$  [FIFO : {Port  $\rightarrow$  Seq(Adr)},
70           WL  : {Port  $\rightarrow$  Seq(Adr)}]
75            $\wedge$  chan  $\in$  [ val: [HLink  $\rightarrow$  Packet  $\cup$  {NoData}], stt: [HLink  $\rightarrow$  {"free", "busy"}]]
77            $\wedge$  TxQ  $\in$  [TxChan  $\rightarrow$  Seq(Packet)]
            $\wedge$  tstate  $\in$  [UTask  $\rightarrow$  {"running", "ready", "wait4anS", "wait4anR"}]
    
```

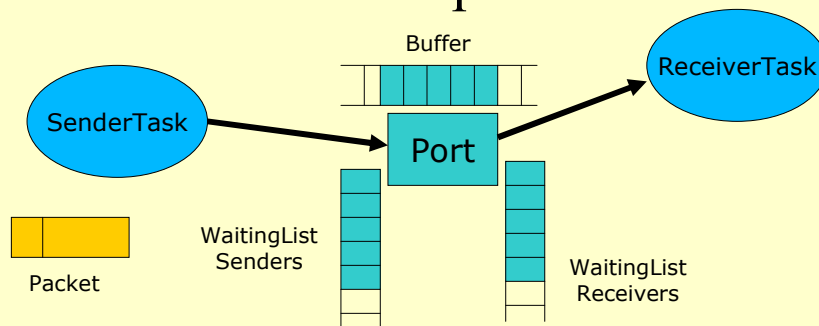
28/06/2007

Open License Society

17



# One result as example



- Need for either FIFO Buffer or WaitingList
  - Both (abstract) models are the same
  - Natural language is imprecise, semantics are context driven
- Benefits:
  - Infinite buffering until no more memory (for Packets)
  - Overflow-free buffering

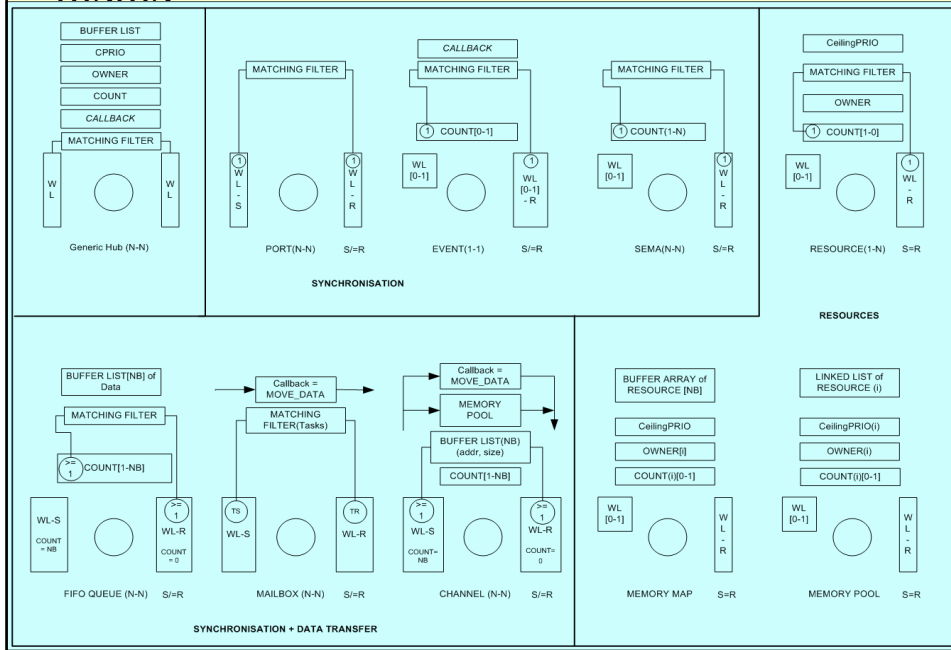
28/06/2007

Open License Society

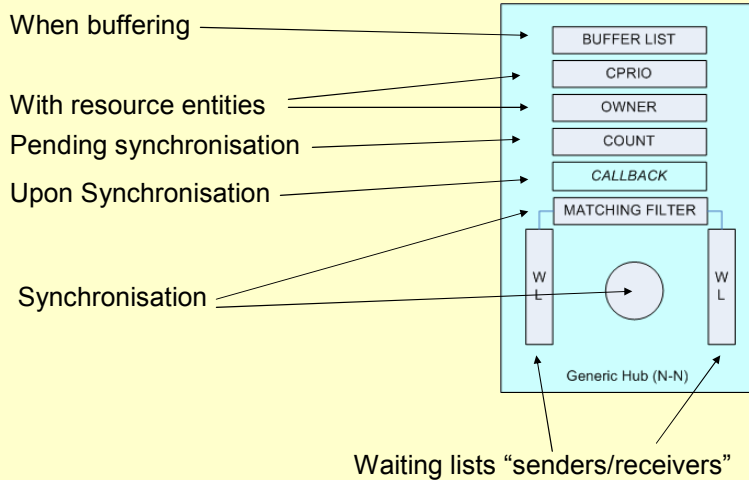
18



# All (typical) RTOS Entities: variations on a theme



# Generic hub: => define your own entities and interactions



# L1 entities

| L1 Entity                 | Semantics                                                                                                   |
|---------------------------|-------------------------------------------------------------------------------------------------------------|
| <b>Event</b>              | Synchronisation on Boolean value. Waiting list on both sides.                                               |
| <b>Counting Semaphore</b> | Synchronisation with counter allowing asynchronous signaling.                                               |
| <b>Port</b>               | Synchronisation with exchange of a Packet.                                                                  |
| <b>FIFO queue</b>         | Buffered communication of Packets. Synchronisation when queue is full or empty.                             |
| <b>Resource</b>           | Event used to create a logical critical section. Resources have an owner Task when locked                   |
| <b>Memory Pool</b>        | Linked list of memory blocks protected with a resource                                                      |
| <b>Mailbox</b>            | Synchronising entity with matching filter on Task ID. Communication happens as side-effect.                 |
| <b>Channel</b>            | Asynchronous communication between Tasks with buffering using memory pools. Communication as a side-effect. |

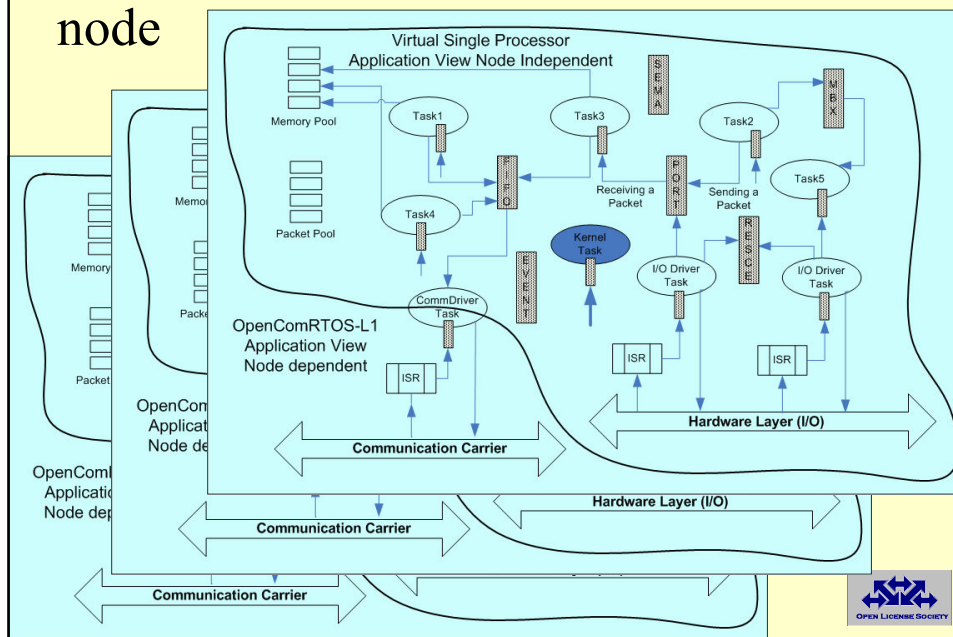
28/06/2007

Open License Society

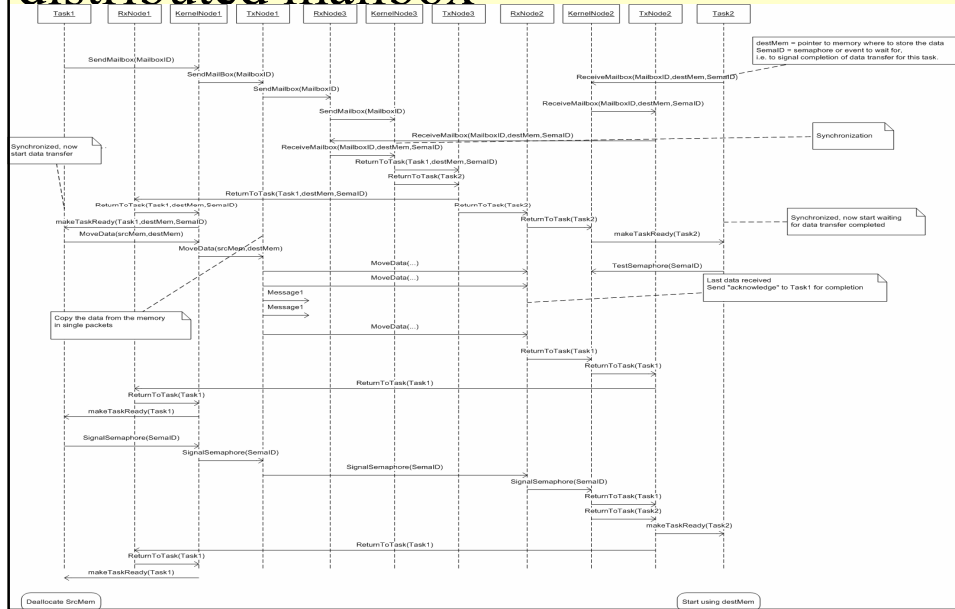
21



any entity can be mapped onto any node



# Example of Interaction diagram: distributed mailbox



## Clean architecture gives small code

OpenComRTOS L1 code size figures (MLX16)

|                          | MP FULL     |             | SP SMALL   |             |
|--------------------------|-------------|-------------|------------|-------------|
|                          | L0          | L1          | L0         | L1          |
| L0 Port                  | 162         |             | 132        |             |
| L1 Hub shared            |             | 574         |            | 400         |
| L1 Port                  |             | 4           |            | 4           |
| L1 Event                 |             | 68          |            | 70          |
| L1 Semaphore             |             | 54          |            | 54          |
| L1 Resource              |             | 104         |            | 104         |
| L1 FIFO                  |             | 232         |            | 232         |
| L1 Resource List         |             | 184         |            | 184         |
| <b>Total L1 services</b> |             | <b>1220</b> |            | <b>1048</b> |
| <b>Grand Total</b>       | <b>3150</b> | <b>4532</b> | <b>996</b> | <b>2104</b> |

Smallest application: 1048 bytes program code and 198 bytes RAM (data)

(SP, 2 tasks with 2 Ports sending/receiving Packets in a loop, ANSI-C

Number of instructions : divide by 2)

## Semantic variations

| Services variants              | Synchronising Behaviour                                                                                                                                                                                    |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>“Single-phase” services</b> |                                                                                                                                                                                                            |
| <b>_NW</b>                     | <b>Non Waiting:</b> when the matching filter fails the Task returns with a RC_Failed                                                                                                                       |
| <b>_W</b>                      | <b>Waiting:</b> when the matching filter fails the Task waits until such events happens.                                                                                                                   |
| <b>_WT</b>                     | <b>Waiting with a time-out.</b> Waiting is limited in time defined by the time-out value.                                                                                                                  |
| <b>“Two-phase” services</b>    |                                                                                                                                                                                                            |
| <b>_Async</b>                  | <b>Asynchronous:</b> when the entity is compatible with it, the Task continues independently of success or failure and will resynchronize later on. This class of services is called “two-phase” services. |

28/06/2007

Open License Society

25



## Classes of services: API

- L0\_Start/Stop/Suspend/ResumeTask
- L0\_SetPriority
- L1\_SendTo/ReceiveFromHub
- L1\_Raise/TestForEvent\_(N)W(T)\_Async
- L1\_Signal/TestSemaphore
- L1\_Send/ReceivePacket L1\_WaitForAnyPacket
- L1\_Enqueue/DequeueFIFO
- L1\_Lock/UnlockResource
- L1\_Allocate/DeallocatePacket
- L1\_Get/ReleaseMemoryBlock
- L1\_MoveData
- L1\_SendMessageTo/ReceiveMessageFromMailbox
- L1\_SetEventTimerList
- ... => user can create his own!

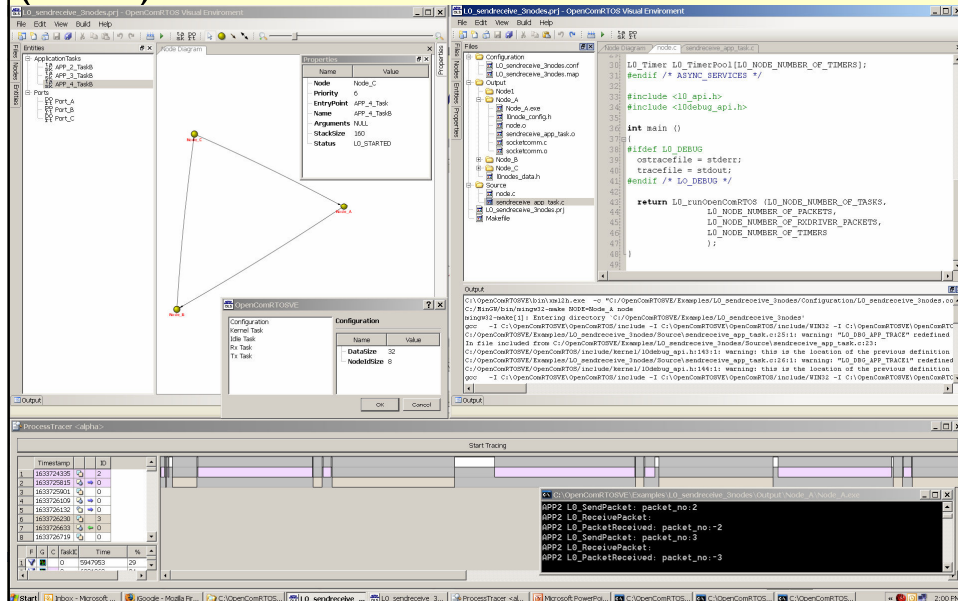
28/06/2007

Open License Society

26



# OpenComRTOS Visual Environment (beta)



## What about real-time scheduling?

- von neumann machine is resource: need to share time: => scheduling
  - Scheduling should be orthogonal to application logic
  - Timer based or priority based but preemptive
  - Priority inheritance mechanism needed, but awkward to implement (code is everywhere)
- Communication backbone is resource: need to share medium
  - issues are latency, P2P bandwidth, buffering
  - => packet switching
  - => priorities inherited

## What about safety?

- Datastructures are passive entities and local
- No buffer overflow, automatic throttling
- Multiple kernel tasks on single node (e.g. supervisor or back-up possible)
  - Software TMR possible, even across nodes
- (most) HW could provide more support
  - Memory corruption
  - Stack space protection
  - Data path bit error detection
  - Recovery points
  - Trustworthy communication backbone crucial => e.g. SpaceWire (IEEE 1355)

28/06/2007

Open License Society

29



## What about security?

- Mostly application level issue
- But:
  - Shuffling pointers hide packet content
  - Data in Packets can be protected/encrypted
  - Packet = memory block with identifier
    - Hashing possible
  - Security supervisor tasks possible
  - Transactions can be secured
  - Matching filter can be enhanced for security (authentication)
- Many topics for future research
  - L2 layer mostly dynamic (occam-Pi ?, Erlang?)
  - Goal: VM < 10 KB

28/06/2007

Open License Society

30



## Results (ctd)

- Break-through results in well-known domain
  - 100's of RTOS with such support
  - 15 years of experience, 3 generations of distributed RTOS design (Virtuoso RTOS – Eonic Systems)
  - Typically CPU dependent, use of assembler and async operation
- Small, scalable, distributed and maintainable code
  - SP(L0): < 1000 machine instructions
  - MP(L1): < 2000 - 5000 machine instructions
  - Needs a few 100 bytes of data RAM
  - Fully in ANSI-C, MISRA-C compliant
  - Runs on MelexCM (16 bit) and Windows, ports underway
  - Scheduling algorithm could be improved to reduce worst-case rescheduling latency and blocking time
  - All RTOS Entities are variations of a generic « hub » object
    - => less but faster code: **5 KBytes vs. 50 KBytes before**
    - **RT performance @ 5 Mips, what needed 50 Mips before**

28/06/2007

Open License Society

31



## Issues with TLA+ / TLC

- Needs a few months to get the right modeling style (especially concurrency)
- TLC declares critical section over all actions
  - In RTOS must be minimal
  - Requires good know-how of target processor
  - Why can't FM not give the minimum critical sections?
- State Space is exponential
  - Millions of states for small application test model
  - TLA model not parametric
  - Might need hours to check
  - Tracing illegal states not always trivial
  - But not useable for checking numerical properties

28/06/2007

Open License Society

32



## Key observations

- Successive iterations: evolutionary
  - > 28 successive models from 2 pages to 25 pages
  - Initially very abstract, neglecting details
  - All successive models were correct, why ?
    - Iterative, incremental process!
    - Takes 15 minutes from one model to the next
- Interaction and abstraction
  - Interplay between SW architects and formal modeling engineer
  - Architectural model polluted by programming concepts
  - Abstraction from TLA helped to find these issues
  - Formalised thinking
- Much cleaner, safer and performant architecture
- Caveat: FM do not prove software is correct (! ?)
  - Proves that Formal **Models** are correct

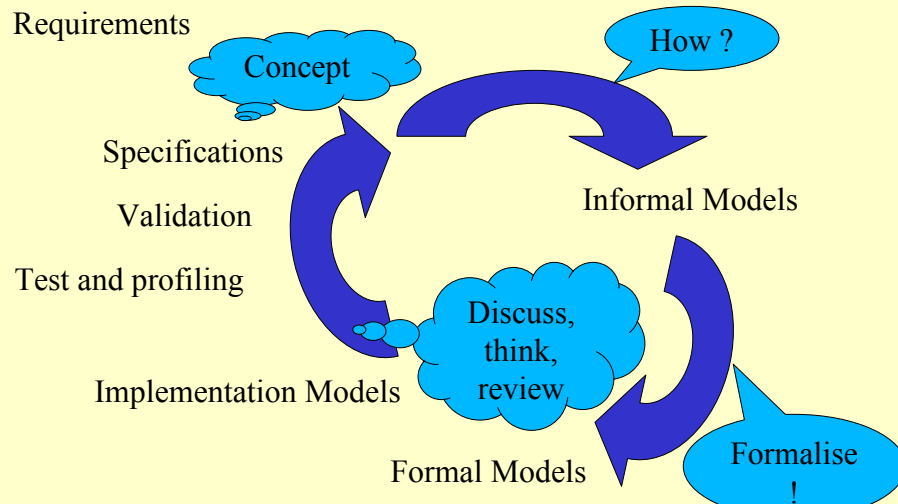
28/06/2007

Open License Society

33



## How it really works: teamwork



28/06/2007

Open License Society

34



# Summary

- Open License Society's approach is about 'formalised thinking'
- The essence is the SE process
  - not the tools, but they help a lot
  - Applying occam's rule: find the minimal solution
- The benefits are "things being done better"
  - OpenComRTOS reinvents the RTOS
  - Smaller, safer, more performant applications
  - Very well suited for multi-core, networked systems
  - Defines a scalable programming methodology
  - Might migrate into the hardware
- Contact:
  - [eric.verhulst@OpenLicenseSociety.org](mailto:eric.verhulst@OpenLicenseSociety.org)

28/06/2007

Open License Society

35

