I-SENSE

Intelligent Multi-Sensor Fusion

Allan Tengg[1], Andreas Klausner[1], Bernhard Rinner[2]

[1] Institute for Technical Informatics
Graz University of Technology
AUSTRIA
{tengg, klausner}@iti.tugraz.at

[2] Institute of Networked and Embedded Sytems
Klagenfurt University
AUSTRIA
bernhard.rinner@uni-klu.ac.at

Presentation Overview

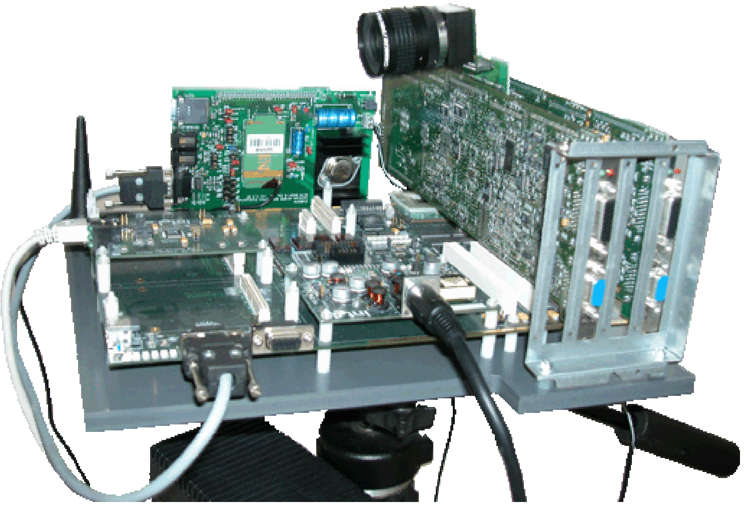
- Motivation
- Modelling of a distributed data fusion system
- Configuration Method
- Middleware Overview
  - System services
  - I-Sense API
- Case Study: Traffic Surveillance
  - Video- and Audio-analysis
  - Support Vector Machines for Data Fusion
- Results
- Conclusion and Outlook

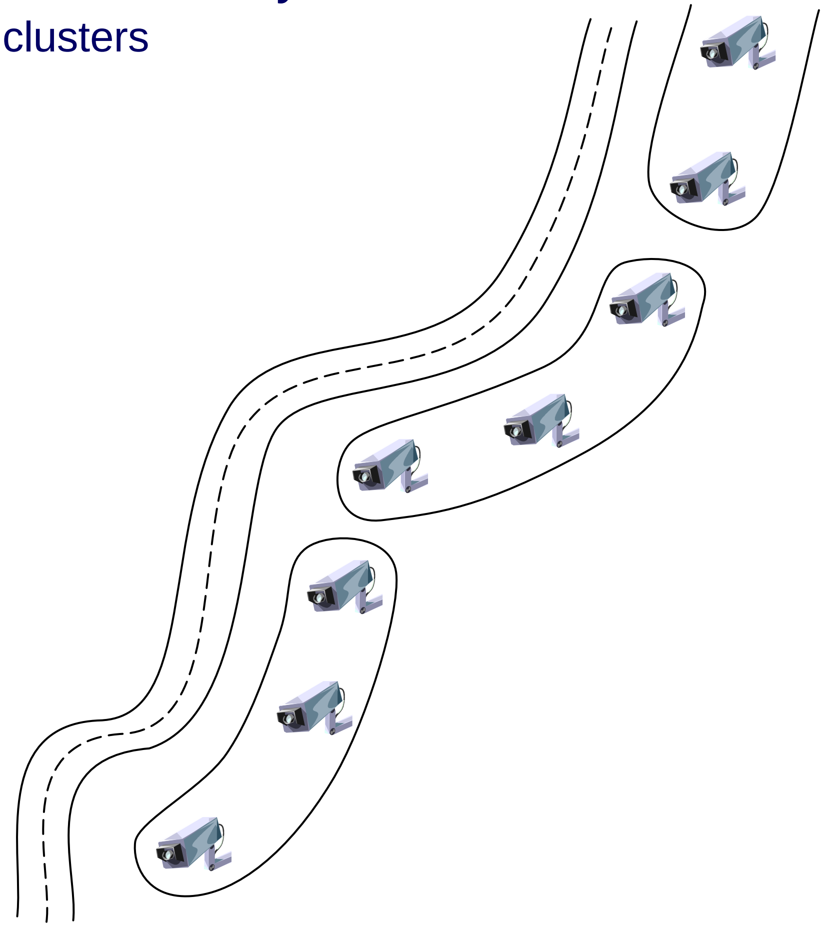
Motivation

- Sensor Data Fusion
  - Extended spatial and temporal coverage
  - Increased robustness and confidence
- Application independent Data Fusion Architecture
- Intelligent Sensor Nodes
  - Embedded Systems with Sensing- and Data Processing Units
- Geographically distributed Sensor Nodes
- **Middleware for Distributed Data Fusion Applications**

Preliminary Projects - *SmartCam*

- Traffic surveillance system based on video only
  - Tracking moving objects in SmartCam clusters
    - Detect collisions/accidents/traffic jam
    - Spot wrong-way driver
    - Find lost cargo
  - MPEG-4 streaming





I-SENSE Hardware Model (1/2)

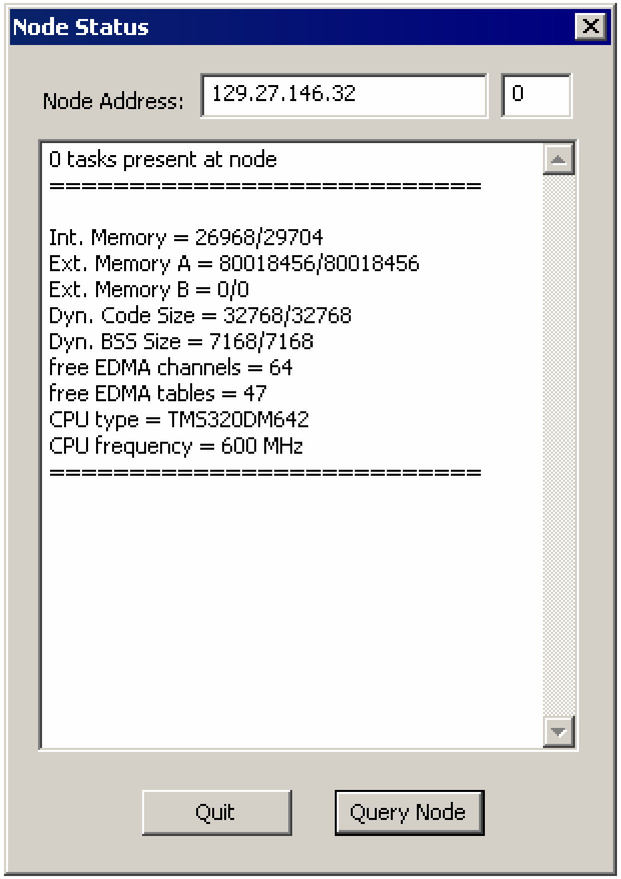
- ➢ General Purpose Processors:
  - ➢ Embedded Pentium Mobile Boards
  - ➢ Up to 1.6 GHz, 512 MB RAM
  - ➢ CF Card for OS and Application
  - ➢ Ethernet, USB, Serial, PCI

- ➢ Digital Signal Processor Cards:
  - ➢ TMS320DM642
  - ➢ 600 MHz, 128 MB
  - ➢ Ports for PAL/NTSC video signals

👍 Substantial processing power

I-SENSE Hardware Model (2/2)

- Sensors
  - Color Cameras
  - Infra-Red Cameras
  - Professional Audiocard and Microphones
  - Light Barriers
- Explore Hardware automatically
  - List of Sensor Nodes
  - Sensor Nodes report their free resources
  - Build and parameterize HW model during
  - Detect faulty or missing nodes

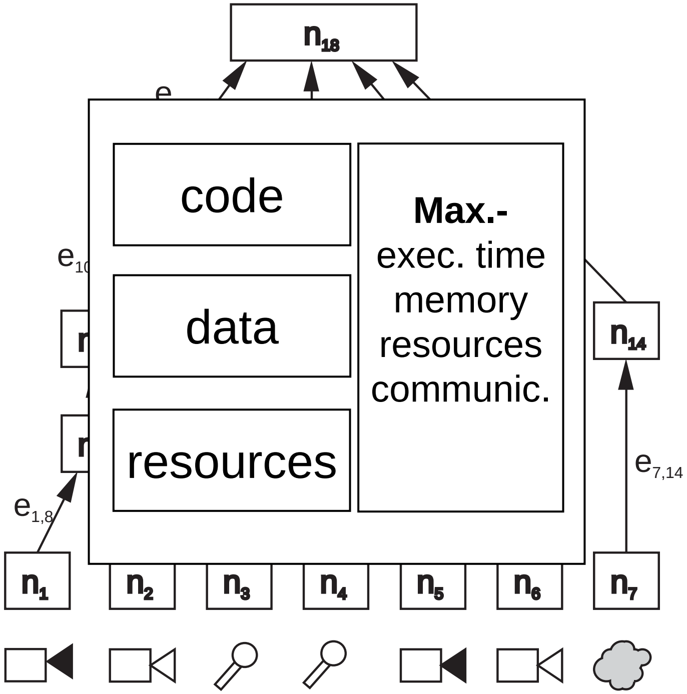


Node Address: 129.27.146.32 0

```
0 tasks present at node
==========================

Int. Memory = 26968/29704
Ext. Memory A = 80018456/80018456
Ext. Memory B = 0/0
Dyn. Code Size = 32768/32768
Dyn. BSS Size = 7168/7168
free EDMA channels = 64
free EDMA tables = 47
CPU type = TMS320DM642
CPU frequency = 600 MHz
==========================
```

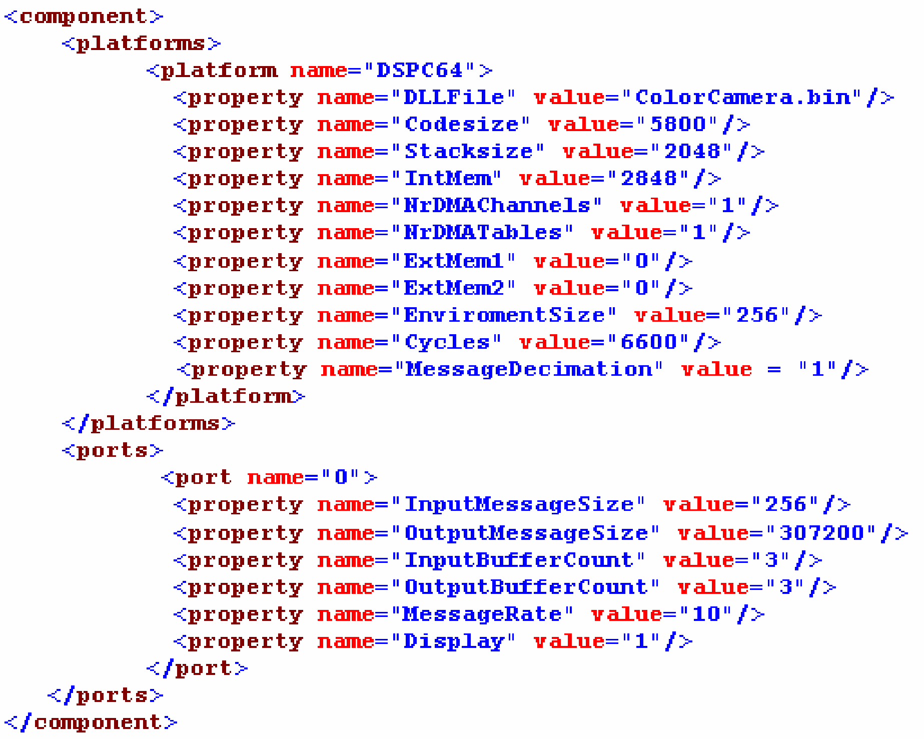
I-SENSE Fusion Model

- Task Graph **G** = (**N**, **E**)
  - **N** = (n_1, n_2, n_m)
  - **E** = (e_{12}, e_{13}, … e_{nm})
- **G** is weighted, directed, acyclic
- Hardware requirements of task n_x
- Communication bandwidth between task n_x and n_y: $e_{x,y}$



Description of Fusion Tasks

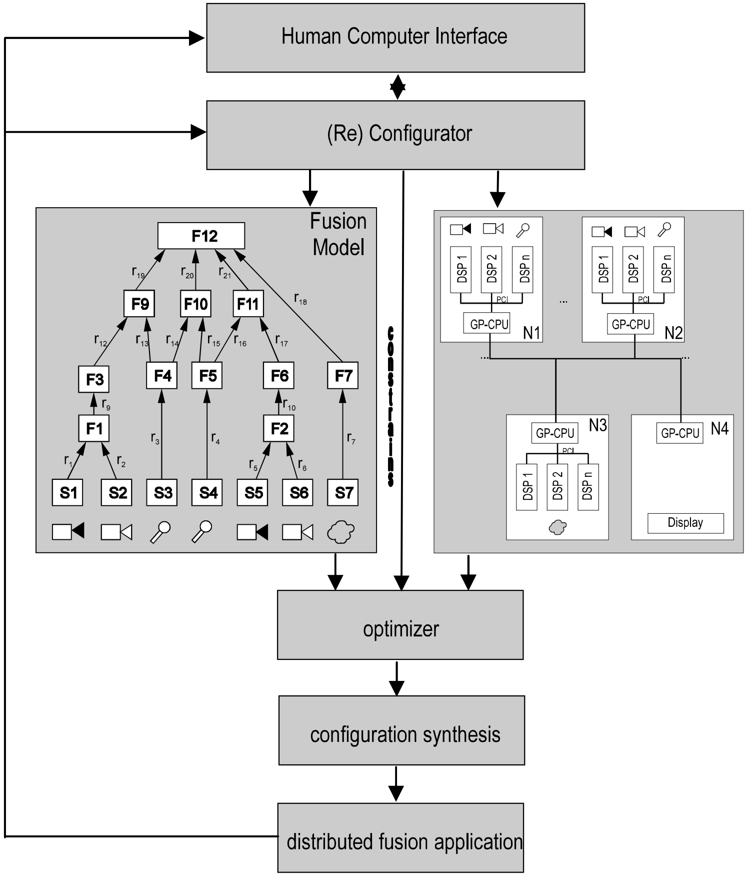
- Fusion Tasks are defined by:
  - Dynamic loadable piece of code (written in C/C++)
  - Tasks Meta information (XML format)



```
<component>
    <platforms>
          <platform name="DSPC64">
            <property name="DLLFile" value="ColorCamera.bin"/>
            <property name="Codesize" value="5800"/>
            <property name="Stacksize" value="2048"/>
            <property name="IntMem" value="2848"/>
            <property name="NrDMAChannels" value="1"/>
            <property name="NrDMATables" value="1"/>
            <property name="ExtMem1" value="0"/>
            <property name="ExtMem2" value="0"/>
            <property name="EnviromentSize" value="256"/>
            <property name="Cycles" value="6600"/>
            <property name="MessageDecimation" value = "1"/>
          </platform>
    </platforms>
    <ports>
           <port name="0">
            <property name="InputMessageSize" value="256"/>
            <property name="OutputMessageSize" value="307200"/>
            <property name="InputBufferCount" value="3"/>
            <property name="OutputBufferCount" value="3"/>
            <property name="MessageRate" value="10"/>
            <property name="Display" value="1"/>
          </port>
   </ports>
</component>
```

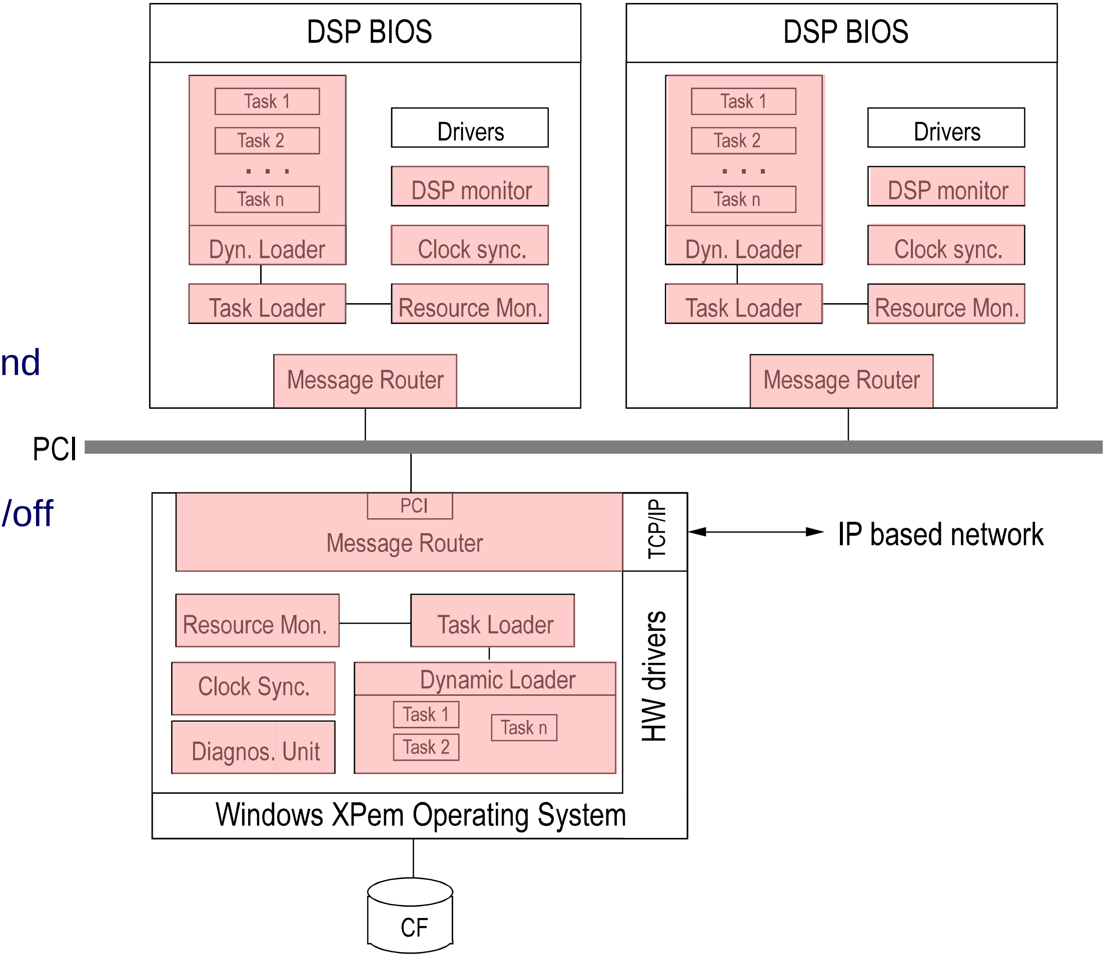
I-SENSE Configuration method

- Initial models provided by the developer
- Optimizer finds valid configuration
- Configuration is loaded on the distributed, embedded system
- Fusion application is running and producing results
- Results are presented on screen
- Operator can modify the Fusion- and Hardware - Model
- Specific events or system faults require on of the models to be altered

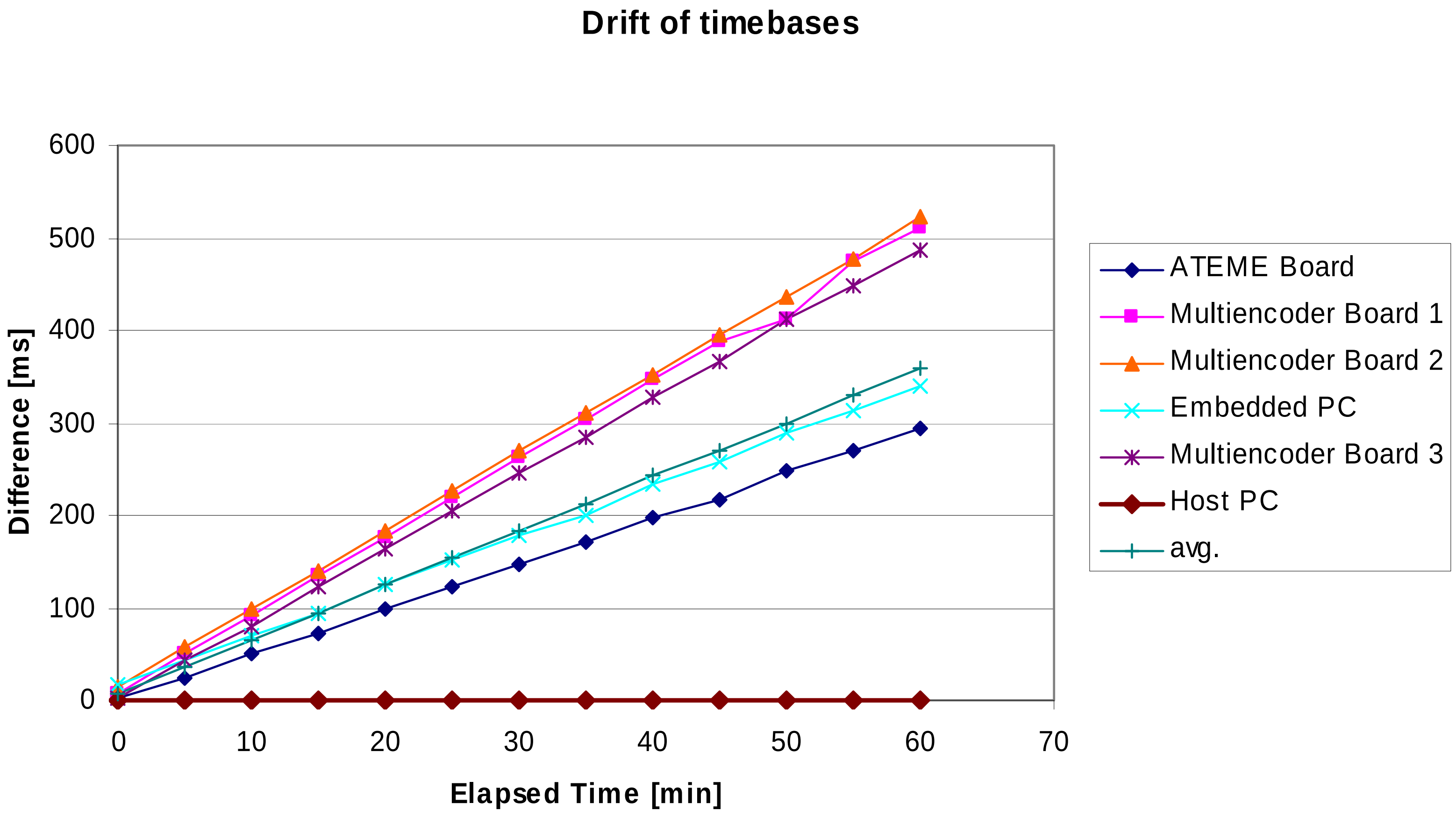


I-SENSE Middleware

- **Message Router**
  provides system and user-data exchange between via PCI and IP network
- **Task Loader**
  Load/Remove/Query tasks on node
- **Dynamic Loader**
  Add/Remove user tasks at runtime and provide API for user tasks
- **Resource Monitor**
  manages all hardware resources (on/off chip memory, DMA channels, …)
- **DSP Monitor**
  determines DSP usage over time
- **Diagnosis Unit**
  check node- and connection- status periodically to detect system failures
- **Clock Synchronization**
  maintains a system-wide timebase

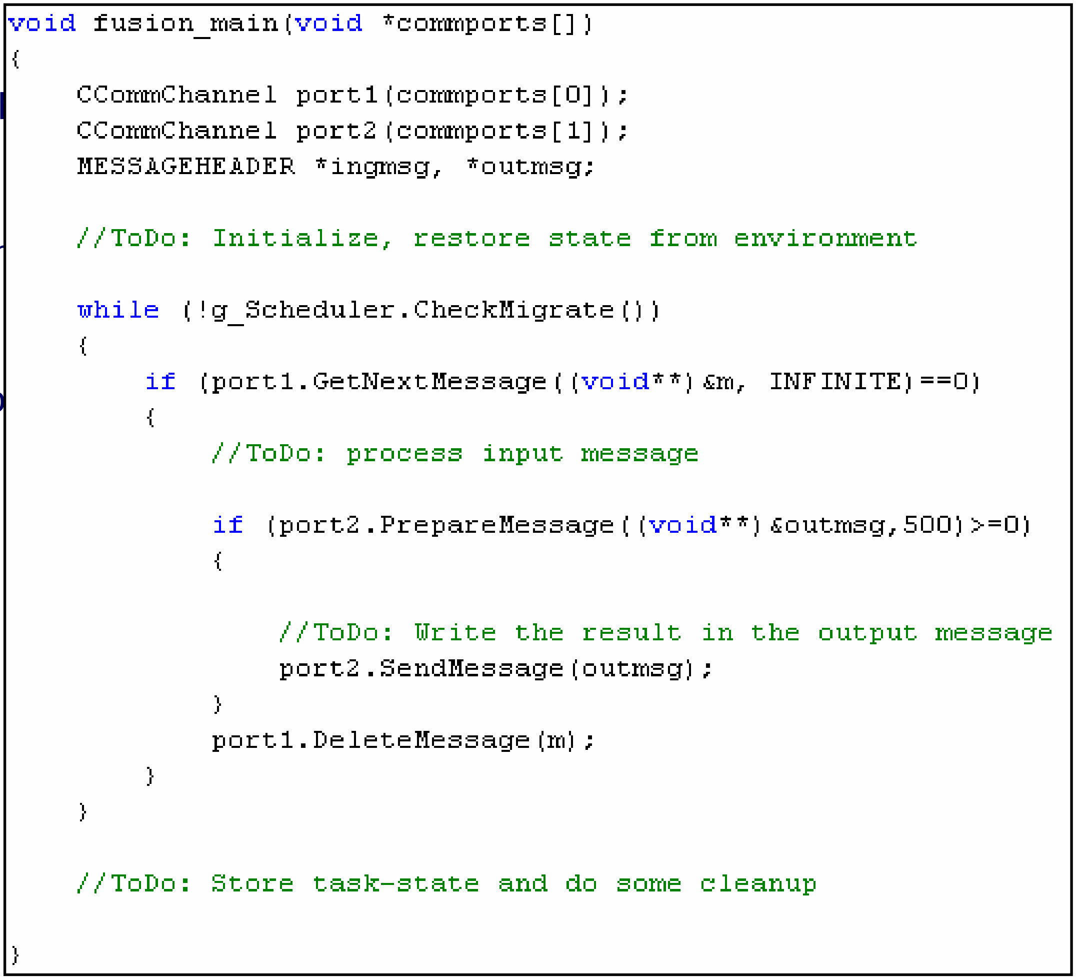


Clock Synchronization



The I-SENSE API

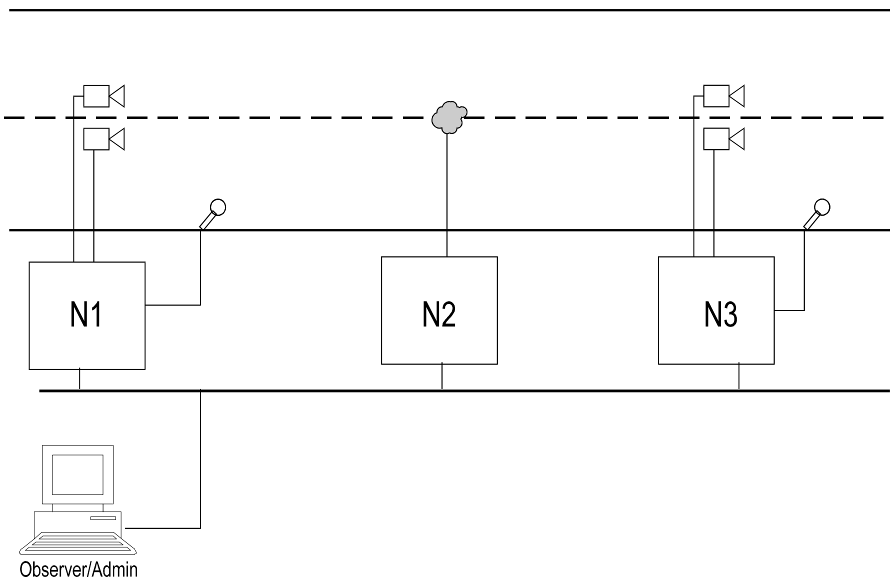
- CommChannel
  - Bidirectional communication with ot
- Timebase
  - Query systemwide synchronized tim
- Memory Manager
  - Hardware independent memory allo and administration
- Scheduler
  - Start/Stop new threads
- DMA Manager
  - Administrate DMA resources
  - Ease DMA usage
- Task Environment
  - Tasks own memory block



```
void fusion_main(void *commports[])
{
    CCommChannel port1(commports[0]);
    CCommChannel port2(commports[1]);
    MESSAGEHEADER *ingmsg, *outmsg;

    //ToDo: Initialize, restore state from environment

    while (!g_Scheduler.CheckMigrate())
    {
        if (port1.GetNextMessage((void**)&m, INFINITE)==0)
        {
            //ToDo: process input message

            if (port2.PrepareMessage((void**)&outmsg,500)>=0)
            {

                //ToDo: Write the result in the output message
                port2.SendMessage(outmsg);
            }
            port1.DeleteMessage(m);
        }
    }

    //ToDo: Store task-state and do some cleanup

}
```

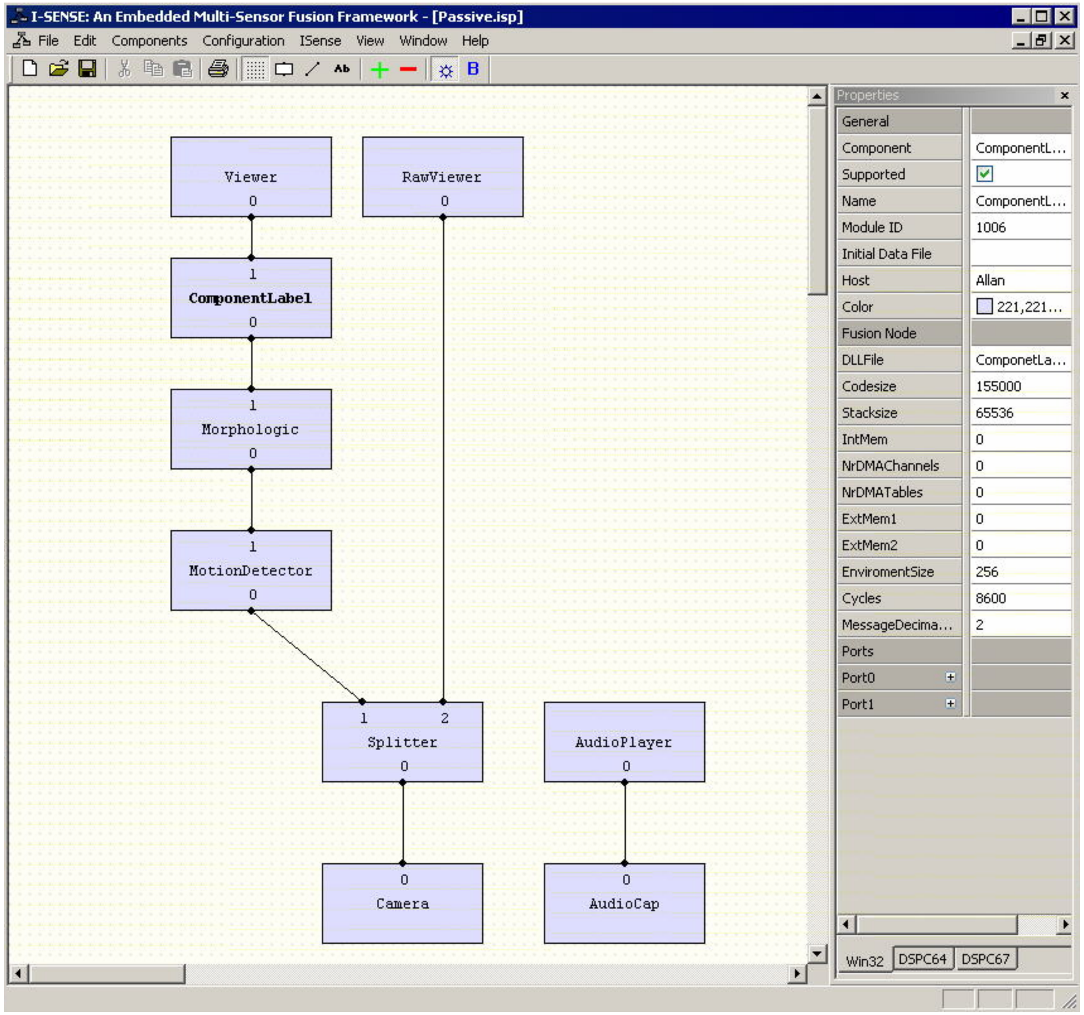
I-SENSE Case Study

- Audio Visual Traffic Surveillance
  - Many distributed sensors
  - High data rates
  - High computation requirements



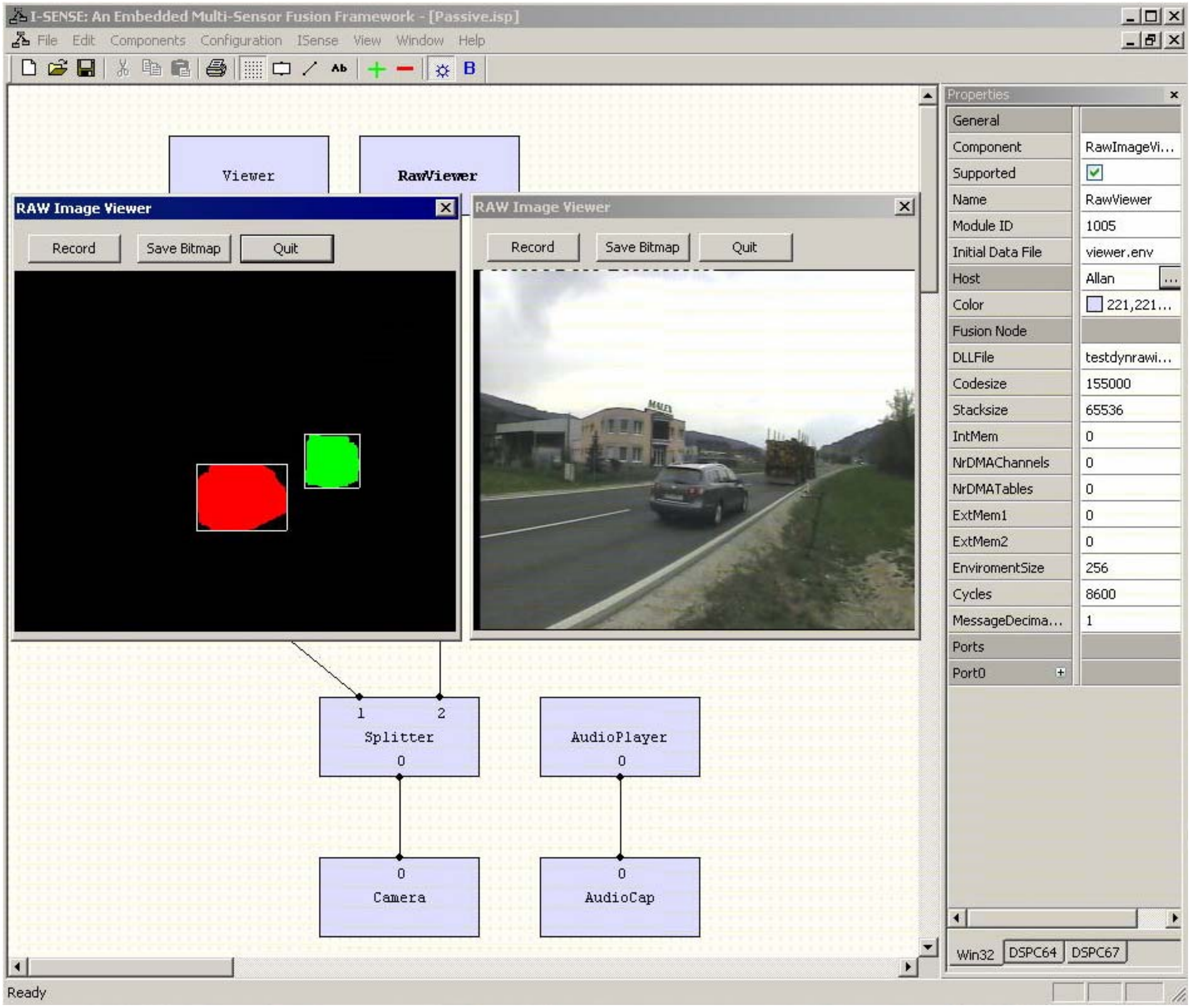
- Sophisticated algorithms
  - Video processing
    - acquisition, filtering, fusion, recording of video streams
    - Motion detection
    - Visual Feature Extraction
  - Audio processing
    - Aquisition, filtering, transformation and recording of audio streams
    - Acoustic object recognition
    - Acoustic feature extraction
  - Object tracking and classification (with SVM)

Defining a distrib. fusion application



General	
Component	ComponentL...
Supported	☑
Name	ComponentL...
Module ID	1006
Initial Data File	
Host	Allan
Color	□ 221,221...
Fusion Node	
DLLFile	ComponetLa...
Codesize	155000
Stacksize	65536
IntMem	0
NrDMAChannels	0
NrDMATables	0
ExtMem1	0
ExtMem2	0
EnviromentSize	256
Cycles	8600
MessageDecima...	2
Ports	
Port0 ⊞	
Port1 ⊞	

Running the application



I-SENSE: An Embedded Multi-Sensor Fusion Framework - [Passive.isp]

General	
Component	RawImageVi...
Supported	☑
Name	RawViewer
Module ID	1005
Initial Data File	viewer.env
Host	Allan
Color	221,221...
Fusion Node	
DLLFile	testdynrawi...
Codesize	155000
Stacksize	65536
IntMem	0
NrDMAChannels	0
NrDMATables	0
ExtMem1	0
ExtMem2	0
EnviromentSize	256
Cycles	8600
MessageDecima...	1
Ports	
Port0	

Ready

Experimental Results (1/2)

- Performance of Middleware: Task loading

Task Name	CPU type	Code Size	Environment	Time
Image Viewer	Pentium	154 kB	256 Byte	139.7 ms
Motion Detector	Pentium	162 kB	512 kByte	273.6 ms
Camera Driver	DSP	5 kB	256 Byte	23.3 ms
Motion Detector	DSP	7 kB	512 kByte	143.6 ms

Experimental Results (2/2)

- Performance of Middleware: Task migration

Task Name	Source	Code Size	Environment	Destination	Time
Image Viewer	Pentium	154 kB	256 Byte	remote Pentium	508 ms
Camera Driver	DSP	5 kB	256 Byte	local DSP	436 ms
Camera Driver	DSP	5 kB	256 Byte	remote DSP	476 ms
Motion Detector	Pentium	7 kB	512 kB	local DSP	520 ms
Motion Detector	Pentium	7 kB	512 kB	remote DSP	552 ms
Motion Detector	DSP	162 kB	512 kB	local Pentium	613 ms
Motion Detector	DSP	7 kB	512 kB	local DSP	395 ms
Motion Detector	DSP	162 kB	512 kB	remote Pentium	623 ms

Conclusion and Outlook

- 👍 Fusion application can be defined quickly
- 👍 Task allocation can be changed while system is running
- 👍 System able to adapt its functionality when needed

- 👎 Application must be halted during reconfiguration
- 👎 Status of fusion tasks gets lost

- ☞ Transform the application gradually into a new one
  (by sending task loading, unloading, migration, … requests in a suitable order)
- ☞ Implement sophisticated error handling mechanisms
- ☞ Extend set of fusion algorithm for case study

Thank you for your attention!

http://www.iti.tugraz.at/isense

References

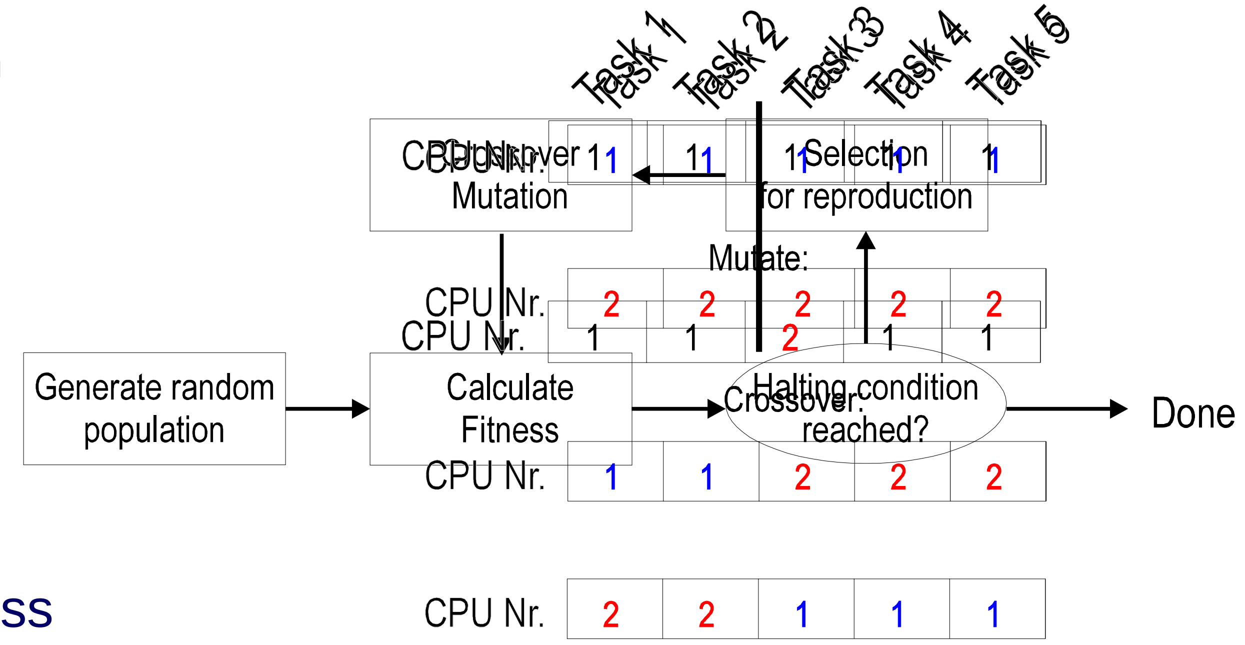
- David L. Hall and James Llinas. Handbook of Multisensor Data Fusion. CRC Press, 2001
- S. S. Bhattacharyya, P. K. Murthy, and E. A. Lee. Software Synthesis from Dataflow Graphs. Kluwer Academic Publishers, 1996
- A. Tengg, A. Klausner, and B. Rinner. An Improved Genetic Algorithm for Task Allocation in Distributed Embedded Systems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2007)*, London, UK, July 2007
- Michael Bramberger, Roman Pflugfelder, Bernhard Rinner, Helmut Schwabach, and Bernhard Strobl. Intelligent Traffic Video Sensor: Architecture and Applications. In *Proceedings of the Workshop on Mobile Computing (TCMC 2003)*, Graz, Austria, March 2003
- H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kulwer Academic Publishers, Norwell, Massachusetts, USA, 1997

Task Allocation with Genetic Algorithm

- **Chromosome**
  possible task allocation
  (m^n possible combinations)

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6		Task n
CPU Nr.	c_2	c_1	c_3	c_2	c_3	c_4	……	c_m

- **Fitness**
  Rating of a task allocation
- **Genetic Operations**
  - Mutation
  - Crossover
- **Genetic Algorithm**



	Task 1	Task 2	Task 3	Task 4	Task 5
CPU Nr.	1	1	1	1	1
CPU Nr.	1	1	1	1	1
CPU Nr.	2	2	2	2	2
CPU Nr.	1	1	2	1	1
CPU Nr.	1	1	2	2	2
CPU Nr.	2	2	1	1	1

- Goal: Optimize Fitness

Scheduling of Fusion Tasks

- Hard Realtime is not possible with the I-SENSE architecture
  - Ethernet components off the shelf not suitable for RT
  - PCI Bus not designed for RT operations
  - Used operating system (Windows XP Embedded)

- Preemtive timeslice scheduler

👍 No schedule needs to be calculated ahead

👍 Every task may begin when all required data are available