Middleware for Dynamic Reconfiguration in Distributed Camera Systems

Milan Jovanovic and Berhard Rinner

Outline

Motivation

Dynamic reconfiguration

Policy-based middleware

Implementation

- Policy structure, Agents and Publisher/Subscriber framework

SmartCam prototype

Dynamic DSP modules and dynamic loading

Results

Motivation

Autonomous dynamic reconfiguration

Network adjustment (better capture the current requirements)

Context analyse

Smart resource utilization, Load distribution, QoS

Policy governed behaviour

Surveillance systems

Distributed surveillance, Large scale surveillance systems

Multiscale surveillance (positions, behaviour, face expressions)

SmartCam : on-board real-time video analysis and streaming

Dynamic Reconfiguration of the Network

State machine

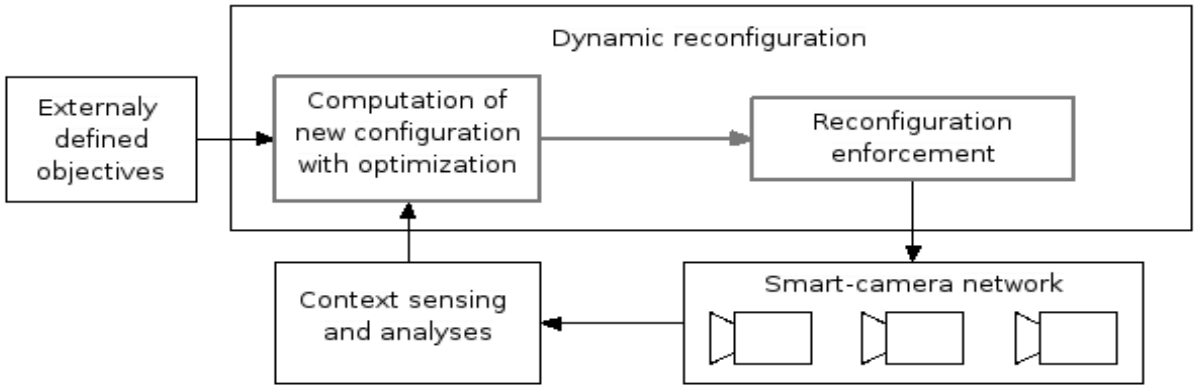
$C = (s \times qs \times r)$

s - services/tasks

qs - service levels

r - resources

$Ci \rightarrow Cj$



Reconfiguration loop

Context sensing and analysis

Computation of new configuration with optimization

Reconfiguration enforcement

Policy-based Framework

Policy-based Framework

A Framework for Policy-based Admission Control

Policy Enforcement Point (PEP)

Application specific interface , Sensors , Events , Actuator

Connections : Request for decision

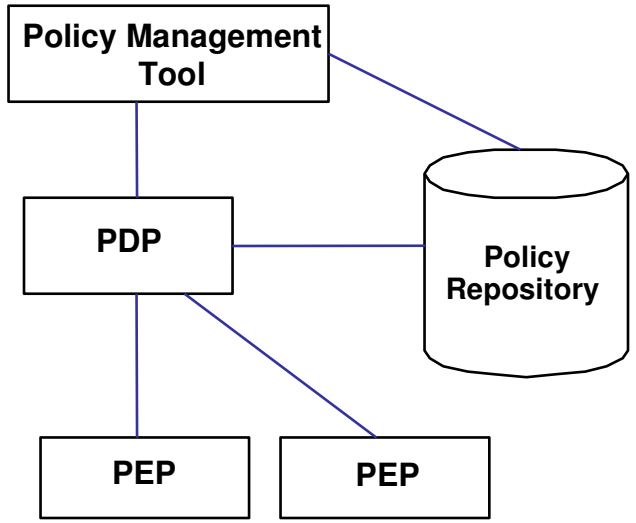
Policy Decision Point (PDP)

Decision , Updates , Policy container

Communication

Message oriented

Protocols : COPS , SNMP ...



Adventage of Policy-based Middleware

Motivation

- Automated management, flexibility , long time adaptation
- Change behaviour without code change

Policy-based administration

- Centralization
  - Policy provisioning
  - Policy configuration
- Business-level abstractions
  - Goal oriented policies (Surveillance scenarios)

Implementation

Configuration layer

- Policy editor (GUI)
  - define new policy / rule / condition / action , edit , reusability
- XML storage (portability between heterogeneous platforms)
- SAX parser

Policy layer

- Policy provisioned
- Decision making process

Executing layer

- Policy outsourced
- Basic actions aware (interface to real functional call / callbacks)

Policy Structure

Data Model

- PCIM paradigm (Policy Core Information Model)
- Object oriented , Nested Structure

Policy

- Sub-policy
- Rule
  - Condition
    - Sub-condition
    - Parameters
  - Action(intern action , basic actions)
    - Sub-actions
    - Parameters (Integer, Long, Boolean, String, Object)

Policy Structure II

Policy processing (decisioning process)

Output : Actions for enforcement

Policy conflict resolution

Priority : specified / inherited from container

Dynamic Policy

Enabling / Disabling sub-policies

Adding / Removing sub-policies

Simplified policy example

POLICY 1	Initialize and start motion detection
RULE 1	If motion detection agent not exist create motion detection agent
CONDITION 1	If motion detection agent not exist
ACTION 1	Create motion detection agent
BASIC ACTION ID (1)	
PARAMETER 3 (ClusterName)	Target cluster
PARAMETER 4 (NodeID) ...	Target node
RULE 2	If motion agent is ready load the algorithm
CONDITION 2	If motion detection agent exist and algorithm is not loaded
OPERATOR (AND)	
CONDITION 1	If motion detection agent exist
CONDITION 3	If motion detection algorithm is not loaded
ACTION 2	Load the algorithm
BASIC ACTION ID (2)	
...	
PARAMETER 6 (AlgorithmLoaded)	Boolean variable with FALSE as default value
RULE 3	If motion detection algorithm is loaded start the algorithm
CONDITION 4	If motion detection algorithm is loaded
ACTION 3	Start the algorithm
...	
PARAMETER 9 (MotionEvent)	Motion event variable with FALSE as default value

Agents

Agent System: Diet Agents

Java on Xscale : JamVM 1.3.0 + GNU Classpath 0.14

Upgraded for distributed environment (service oriented)

Distributed CSP solver (merging the distributed solutions).

Code migration : Message oriented / Agent migration / Object serialization

Policy Agent

PDP / PEP container

Local / Distributed decision

Policy governed behaviour

Agent Hierarchy : Decisioners , Action-enforcers

Publisher/Subscriber Framework

DSP Framework

Runs on every DSP

Abstraction of the hardware and communication channels

Dynamic loading (Texas instruments dynamic loader support)

Xscale Framework

Client/Server architecture

Callback interface

C++ implementation

SmartCam prototype

Intel IXDP425 Baseboard

Intel IXP425 Network Processor

533 MHz, 256 MB SDRAM, 2x

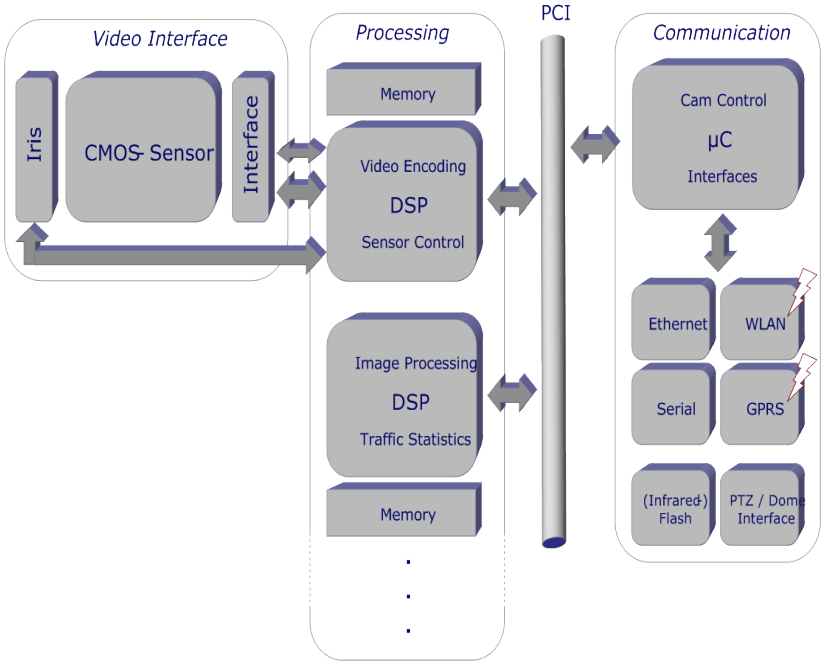
100BaseT Ethernet, PCI

ATEME IEKC64 (2x)

Texas Instruments TMS320C6416

600 MHz, 264 MB SDRAM

CMOS image sensor



Dynamic DSP Modules

Motion detection

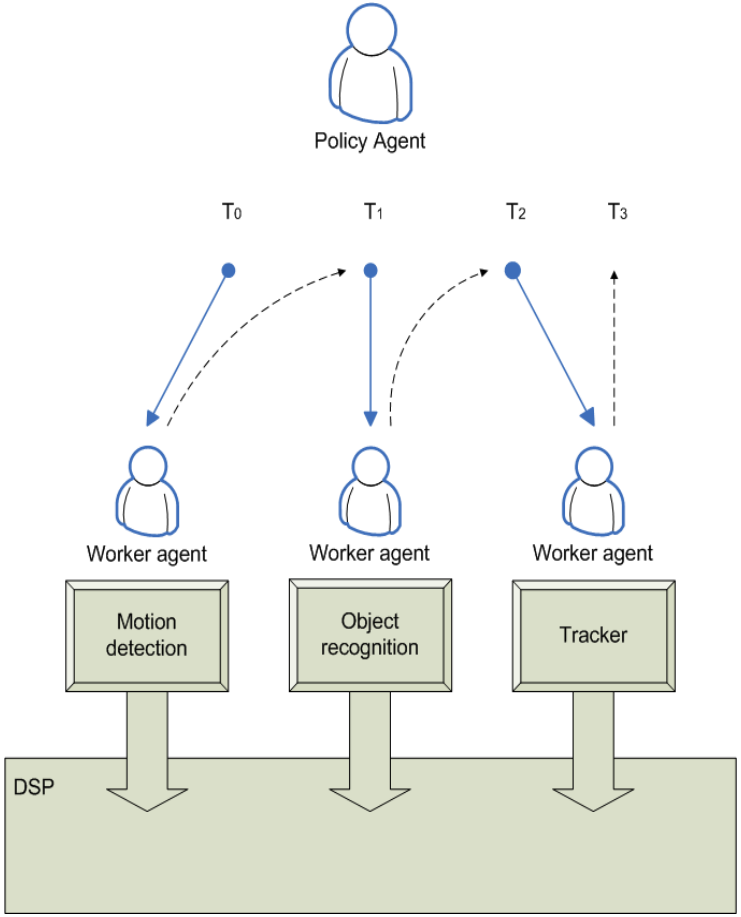
- Background subtraction algorithms
- Temporal difference
- Kalman filter
- Temporal variance

Object recognition

- Object recognition
- Classification

Tracking

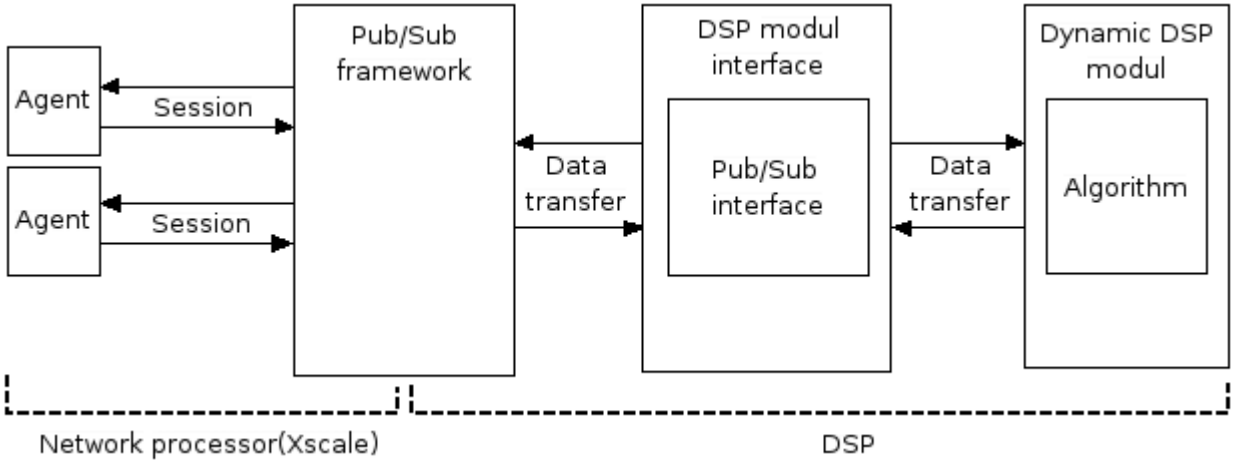
- Object tracker



Dynamic DSP Modules

Interface to mobile agent

- Specific PEP implementation
- Java native interface
- Session with DSP module



Dynamic Loading

Runtime software exchange on DSPs

Scheduling cases

1. Module initialization

   Standby mode

   Memory not critical

2. Loading and initialization of new module

   Unknown module type

3. Unloading , loading and initialization of new module

   Memory deficiency

Results

1. Loading times of dynamic modules dependent on the scheduling case
2. Loading times of dynamic modules dependent on the module size
3. Resource requirements for motion detectors

Scheduling case	$T_{EnforcementEnd} - T_{EnforcementStart}[ms]$				
Case 1	166	201	198	200	199
Case 2	255	246	226	215	192
Case 3	425	412	397	381	360

Module size	$T_{LoadingEnd} - T_{LoadingStart}[ms]$				
5.6 kB	48	45	43	41	42
10 kB	55	58	49	55	58
262 kB	165	167	177	165	170

Algorithm	Processing time [processor cycles]	Required memory[kB]
Temporal Differencing	5.2×10^6	198
Kalman filter	8×10^6	297
Single Gaussian	10×10^6	396

Conclusion and Future work

Evaluation

- Case study (different surveillance scenarios)

Future work

- Optimisation methods
  - Caching of processed policies (Memoization)

Questions