TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Resource management for media processing in networked embedded systems : proceedings of a one-day workshop, Eindhoven, March 31, 2005

Citation for published version (APA):
Bril, R. J., & Verhoeven, R. (Eds.) (2005). *Resource management for media processing in networked embedded systems : proceedings of a one-day workshop, Eindhoven, March 31, 2005*. Technische Universiteit Eindhoven.

Document status and date:
Published: 01/01/2005

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

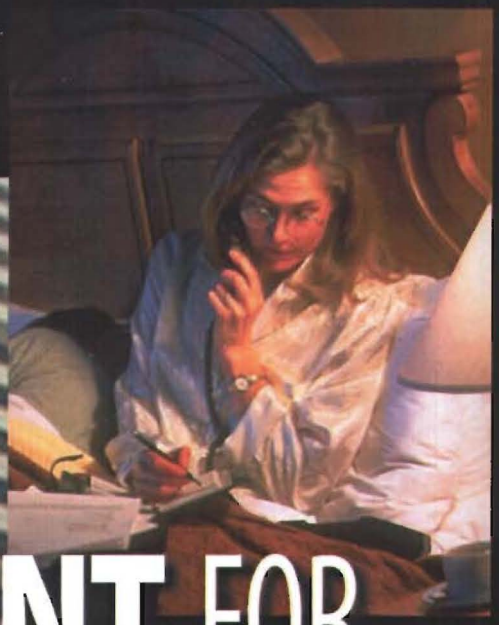
providing details and we will investigate your claim.

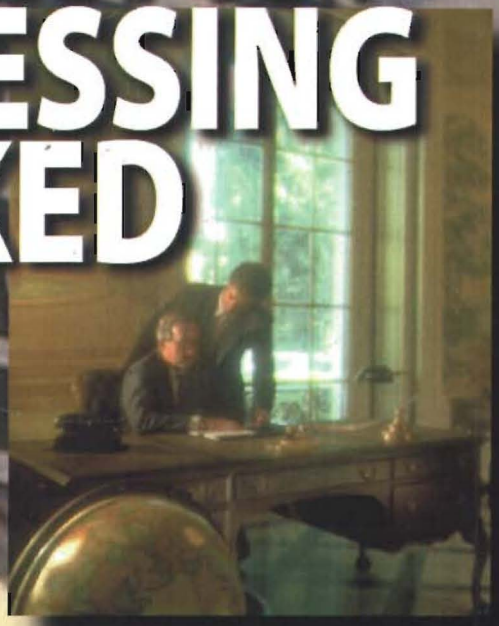
Download date: 04. Oct. 2023

DGF
2005
RES

Proceedings 2005

WORKSHOP ON RESOURCE MANAGEMENT FOR MEDIA PROCESSING IN NETWORKED EMBEDDED SYSTEMS





IEEE Benelux Chapter on Consumer Electronics

Resource management for media processing in networked embedded systems

Reinder J. Bril and Richard Verhoeven (eds.)

Proceedings of a one-day workshop organized by the

IEEE Benelux Chapter on Consumer Electronics

in conjunction with the

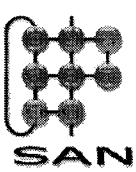
Mathematics and Computer Science Department of the Technische Universiteit Eindhoven

March 31st, 2005

Sponsors

Primary sponsor

Secondary sponsor



Supporting sponsor

TU/e

Copyright © 2005 by the authors. Considerable parts of this text have been or will be published by the IEEE or related institutes.

All rights reserved. No part of this publication may be stored in a retrieval system, transmitted or reproduced in any form or by any means, including but not limited to photography, magnetic, or other record, without prior agreement and written permission of the respective authors.

CIP- DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Resource management for media processing in networked embedded systems: proceedings of a one-day workshop, Eindhoven, March 31st, 2005 / ed. by Reinder J. Bril and Richard Verhoeven. –
Eindhoven: Technische Universiteit Eindhoven, 2005.
ISBN: 90-386-0544-7
NUR 992
Subject headings: resource management / multimedia : proceedings / embedded systems : proceedings
CR Subject Classification (1998): C.4, C.3, C.1, I.4, I.6

Contents

Lectures

Short papers

Earlier releases in this series:

- Proceedings Workshop on "Embedded Video Streaming Technology (MPEG-4) and the Internet", IEEE Benelux Chapter on Consumer Electronics, ISBN 90-386-0991-4, P.H.N. de With (Ed), Technische Universiteit Eindhoven, The Netherlands, December 2001 (155 pages).
- Proceedings Workshop on "The Design of Multimedia Architectures", IEEE Benelux Chapter on Consumer Electronics, ISBN 90-386-0822-5, P.H.N. de With (Ed), Technische Universiteit Eindhoven, The Netherlands, December 2003 (136 pages).

Preface

The IEEE Chapter on Consumer Electronics in the Benelux was founded in the late nineties to support events that are related to applications of Consumer Electronics, and is part of the intemational IEEE CE Society. The CE domain is growing yearly, due to the continuous advances in technology in the area of computing, communication and storage.

The first workshop of the Benelux CE Section was devoted to multimedia video coding for Internet applications. The MPEG video compression standards have been a phenomenal success for the recording and digital distribution of video signals. From these standards, MPEG-2 is most widely applied (e.g. DVD) and MPEG-4 is studied for e.g. portable applications of video systems. The widely accepted use of communication in computer networks is gradually becoming part of the consumer electronics area, leading to communicating consumer video over the internet. This was the theme of the first workshop.

The second workshop of the Benelux CE Section was devoted to the design of multimedia architectures, motivated by the ever increasing density of transistors in a chip. This development poses system designers with the challenge to deal with very complex and divers architectures inside a single system. Given this growing complexity, many CE manufacturers outsource the design of subsystems. The system design owner should subsequently solve the problem of smooth integration and operation of the various subsystems. This complexity control problem occurs both in software and hardware design.

Media processing is often characterized by highly fluctuating, content dependent, resource requirements. Combined with their real-time constraints, media processing puts high demands on resource management in networked embedded systems. This is especially true for consumer systems that provide high-quality media, which have a low tolerance for artifacts and quality fluctuations. The above considerations have led to the theme of this third workshop, which is organized by the Benelux CE Section in conjunction with the Mathematics and Computer Science Department of the Technische Universiteit Eindhoven.

The first lecture, which is given by Prof.dr. Emile H.L. Aarts, explains about the world of ambient intelligence and defines its key characteristics. Prof. Aarts is with Philips Research and the Technische Universiteit Eindhoven, and launched the concept of ambient intelligence. Distributed media applications and their processing on embedded stationary and mobile platforms play a major role in the realization of ambient intelligent environments. We are happy to leam the requirements for the design of ambient intelligent systems and the resulting research challenges from Prof. Aarts.

We are particularly honored that Prof.dr. Christian Hentschel from the University of Technology, Cottbus, Germany, will address scalable video algorithms (SVAs) for resource constrained platforms. Prof. Hentschel is involved in digital video signal processing with a focus on quality improvements since 1989. In his lecture, he will address typical issues on quality for resource-quality SVAs, including proposals for high-quality image processing.

Intelligent control for scalable video processing is presented by Drs. Clemens Wüst and Dr.ir. Wim Verhaegh, who are both with Philips Research in Eindhoven. Video processing in software is often characterized by highly fluctuating, content dependent processing times, and a limited tolerance for deadline misses. In this lecture, they present an approach that

allows close-to-average-case resource allocation to a single video processing task, based on asynchronous, scalable processing, and QoS adaptation.

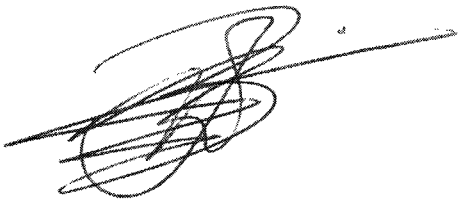
Dr. Peter v.d. Stok presents the fourth lecture of this workshop. He is with Philips Research and the Technische Universiteit Eindhoven. In this lecture, he reports on video streaming over wireless media, such as IEEE 802.11a, 802.11b, and IEE 802.11g, which are very sensitive to perturbations. First, the effects of these perturbations on the video quality are shown. Next, it is shown how a controlled adaptation of the video stream at the sender side reduces the effect of these transmission perturbations.

The final lecture is given by Dr. Wolfgang Eberle from IMEC and the KU Leuven. In this lecture, he reports on design concepts for software-defined and software-reconfigurable radios. These radios are required to adapt to changing quality-of-service demands of the user and to a dynamic environment with limited energy resources. Dr. Eberle illustrates recent cross-layer and mixed-signal design concepts and their successful application in the context of wireless LAN, using MPEG-4 video streaming serves as driver application.

Next to these lectures, the workshop has two sessions with short presentations. These sessions have a work-in-progress like status, in which Ph.D. students present the challenges they address in their research, the directions they are currently pursuing, and initial results. We are very pleased to be able to offer you five state-of-the-art presentations, with topics covering models of the execution of streaming applications, approaches dealing with and user-perception consequences of transmission perturbations in wireless networks, and media processing issues for a multiprocessor system on chip. These presentations are complemented with short papers, which went through a formal reviewing procedure.

The IEEE Benelux Chapter on Consumer Electronics and the Mathematics and Computer Science Department of the Technische Universiteit Eindhoven (TU/e) are pleased to offer this workshop and the enclosed topics to a wide audience. They gratefully acknowledge the System Architecture and Networking (SAN) group of the TU/e for sponsoring this workshop, and PROGRESS, the Embedded Systems Institute, SAI/Software Technology, and IPA for their support.

These proceedings contain a mixture of slide copies and papers addressing the themes of the individual lectures and short presentations. This mixed approach was chosen to give maximum flexibility to the authors with minimum effort, thereby allowing the input of the latest material.



Peter H.N. de With

Board member IEEE Benelux Chapter on CE
Professor Video Coding and Architectures,
Electronics Engineering Faculty,
Technische Universiteit Eindhoven,
The Netherlands

Reinder J. Bril

Assistant Professor,
Mathematics and Computer Science Dept.,
Technische Universiteit Eindhoven,
The Netherlands

Program of "Resource Management for Media Processing in Networked Embedded Systems" (RM4NES)

One-day workshop at the Technische Universiteit Eindhoven (TU/e), The Netherlands, on March 31st, 2005. Organized by the IEEE Benelux Section on Consumer Electronics.

Organization committee

Prof.dr. Emile H.L. Aarts (Philips Research, TU/e)
Prof.dr. Peter H.N. de With (TU/e, LogicaCMG)
Dr. Johan J. Lukkien (TU/e), and
Dr.ir. Reinder J. Bril (TU/e).

Assisting program committee member:
Dr.ir. Richard Verhoeven (TU/e).

Workshop Program

09.00 – 09.35 hrs	Registration and coffee
09:35 – 09:45 hrs	Opening workshop, Dr. Johan J. Lukkien (TU/e)
09:45 – 10:30 hrs	Prof.dr. Emile H.L. Aarts (Philips Research, TU/e) "Ambient Intelligence and System Design"
10:30 – 11:15 hrs	Prof.dr.-ing.habil Christian Hentschel (University of Technology Cottbus, Germany) "Scalable video algorithms for resource constrained platforms"
Break	
11:45 – 12:30 hrs	Drs. Clemens Wüst and Dr.ir. Wim Verhaegh (Philips Research) "Intelligent Control for Scalable Video Processing"
Lunch	
13:45 – 14:30 hrs	Dr. Peter D.V. v.d. Stok (Philips Research, TU/e) "Video Streaming over Home Network"
14:30 – 15:30 hrs	Short presentations: Part I
Break	
16:00 – 16:45 hrs	Short presentation: Part II
16:45 – 17:30 hrs	Dr. Wolfgang Eberle (IMEC and KU Leuven, Belgium) "Quality of experience in wireless multi-standard multi-media devices: a design-time/run-time cross-layer approach"
17:30 – 17:45 hrs	Closing, Prof. dr. Peter H.N. de With (TU/e, LogicaCMG)

Short presentations: Part I

14:30 – 14:50 hrs M.A. Weffers-Albu
"A characterization of streaming applications execution"

14:50 – 15:10 hrs T. Lennvall
"Providing adaptive QoS in wireless networks by traffic shaping"

15:10 – 15:30 hrs D. Jarnikov
"Influence of network awareness on perceived video quality"

Short presentations: Part II

16:00 – 16:20 hrs C.M. Otero Pérez
"Integrating cache space reservations with processing cycles reservations for a multiprocessor system on chip"

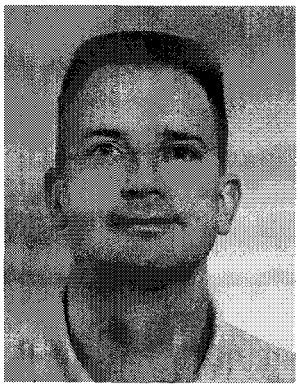
16:20 – 16:40 hrs M. Pastrnak
"Hierarchical QoS concept for multiprocessor system-on-chip"

Contributors

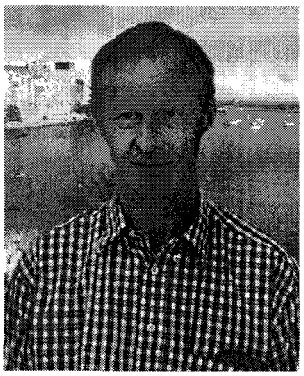
Emile H.L. Aarts is Vice President and Scientific Program Director of the Philips Research Laboratories Eindhoven, The Netherlands. He holds an MSc. and PhD. degree in physics. For almost twenty years he has been active as a research scientist in computing science. Since 1991 he holds a teaching position at the Eindhoven University of Technology as a part-time professor of computing science. He also serves on numerous scientific and governmental advisory boards. He holds a part-time position of senior consultant with the Center for Quantitative Methods in Eindhoven, The Netherlands. Emile Aarts is the author of five books and more than hundred and forty scientific papers on a diversity of subjects including nuclear physics, VLSI design, combinatorial optimization and neural networks. In 1998 he launched the concept of Ambient Intelligence and in 2001 he founded Philips' HomeLab. His current research interests include embedded systems and interaction technology.

Christian Hentschel received his Dr.-Ing. (Ph.D.) in 1989 and Dr.-Ing. habil. in 1996 at the University of Technology in Braunschweig, Germany. He worked on digital video signal processing with focus on quality improvement. In 1995, he joined Philips Research in Briarcliff Manor, USA, where he headed a research project on moiré analysis and suppression for CRT based displays. In 1997, he moved to Philips Research in Eindhoven, The Netherlands, leading a cluster for Programmable Video Architectures. Later he held a position of a Principal Scientist and coordinated a project on scalable media processing with dynamic resource control between different research laboratories. In 2003, he became a full professor at the Brandenburg University of Technology in Cottbus, Germany. Currently he chairs the department of Media Technology. In 2005 he received , together with his co-authors, the Chester Sall Award from the IEEE Consumer Electronics Society for the first place transaction paper. He is a member of the Technical Committee of the International Conference on Consumer Electronics (IEEE) and a member of the FKTG in Germany.

Clemens C. Wüst received the M.Sc. degree in mathematics with honors from the University of Groningen, The Netherlands. Since then, he has been with the Philips Research Laboratories in Eindhoven, The Netherlands, where he has been working mainly on QoS for resource-constrained real-time systems using stochastic optimization techniques. Currently, he is pursuing a Ph.D. degree at the Technische Universiteit Eindhoven.



Wim F.J. Verhaegh received the mathematical engineering degree with honors in 1990 from the Technische Universiteit Eindhoven, The Netherlands. Since then, he is with the Philips Research Laboratories in Eindhoven, The Netherlands. From 1990 until 1998, he has been a member of the department Digital VLSI, where he has been working on high-level synthesis of DSP systems for video applications, with the emphasis on scheduling problems and techniques. Based on this work, he received a Ph.D. degree in 1995 from the Technische Universiteit Eindhoven. Since 1998, he is working on various optimization aspects of multimedia systems, networks, and applications. On the one hand, this concems application-level resource management and scheduling, for optimization of quality of service of multimedia systems. On the other hand, this concerns adaptive algorithms and machine learning algorithms for user interaction issues, such as content filtering and automatic playlist generation.



Peter D.V. van der Stok (1949) holds a M.Sc (1973) and a Ph.D. (1982) degree in physics from the University of Amsterdam. He has worked from 1973 till 1989 at CERN where he became responsible for the operating system software of the distributed control system of the LEP accelerator. He worked from 1989 at Eindhoven University where he is associate professor. His main interests are distributed fault-tolerant real-time systems. Since September 2000 he works for 90% of his time at Philips Research on the subject of In-Home Digital Networks.

Wolfgang Eberle received the M.S. degree in Electrical Engineering from Saarland University, Saarbruecken, Germany, in 1996 with specialization in microwave engineering, telecommunication networks, and biomedical engineering. He joined IMEC in 1997 as a researcher on algorithms and digital VLSI design for OFDM-based wireless LAN. As senior researcher, he was involved in digital compensation techniques for radio front-ends and mixed-signal design methodology. From 2003 on, he headed the research team on mixed-signal and cross-layer design technology. He now leads a systems design team for QoS- and energy-aware systems design and design methodologies for wireless multi-standard multi-media. During his career, he was also affiliated with Fraunhofer IBMT as research assistant, and with the start-up's CardioMed (D) and Resonext Communications (USA; now RFMD) as consultant. Wolfgang Eberle is author or co-author of more than 30 scientific publications.

Ambient Intelligence and System Design

Prof.dr. Emile H.L. Aarts

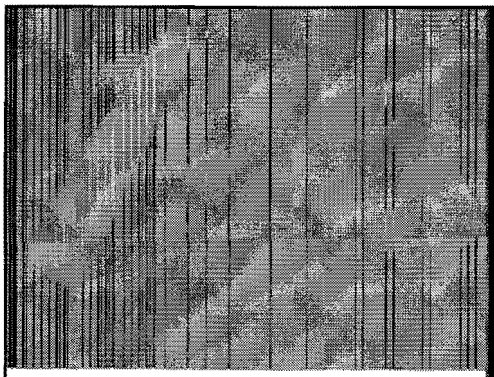
Philips Research, TU/e

Ambient intelligence opens a world of unprecedented experiences: the interaction of people with electronic devices will change as context awareness, natural interfaces, and ubiquitous availability of information will be realized. Distributed applications and their processing on embedded stationary and mobile platforms play a major role in the realization of ambient intelligent environments. Notions as media at your fingertips, enhanced-media experiences, and ambient atmospheres denote novel and inspiring concepts that are aimed at realizing specific user needs and benefits such as personal expression, social presence, and well-being that seem quite obvious from a human perspective, but are quite hard to realize because of their intrinsic complexity and ambiguity. Obviously, the intelligence experienced from the interaction with ambient intelligent environments will be determined to a large extent by the software executed by the distributed platforms embedded in the environment, and consequently, by the algorithms that are implemented by the software.

In the presentation we explain about the world of ambient intelligence and define its key characteristics. Starting from the vision we identify a number of requirements for the design of ambient intelligent systems and the research challenges that result from these requirements in systems design.

Imagineering the future

Emile Aarts
Philips Research

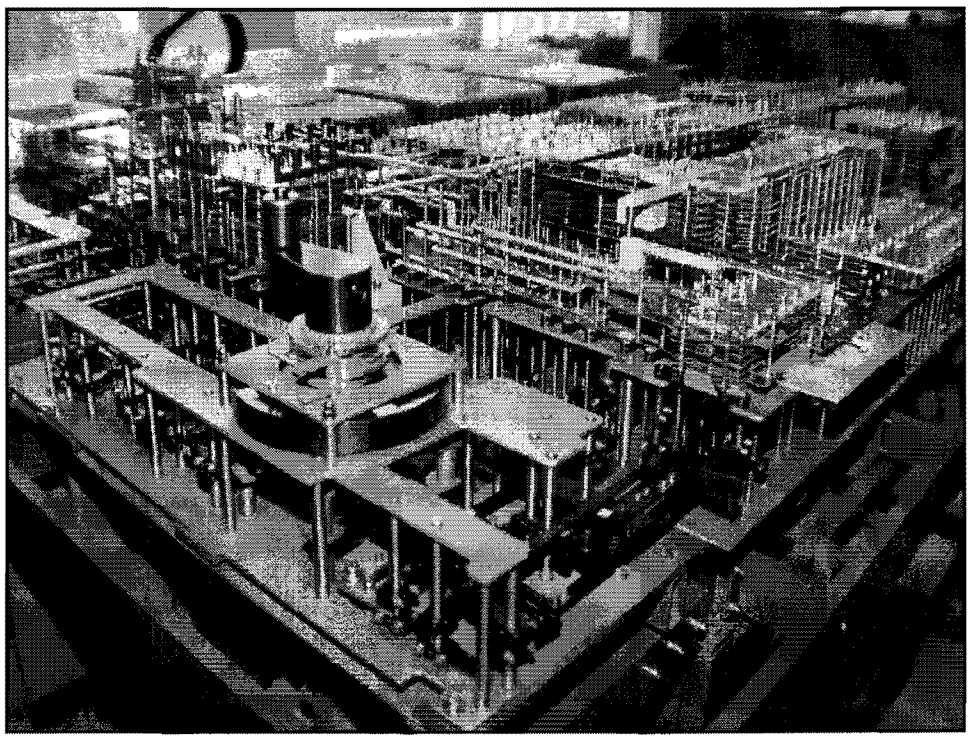


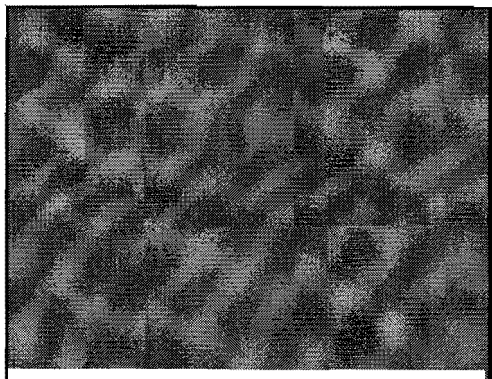
Ambient Intelligence

A progress report

Creating insight

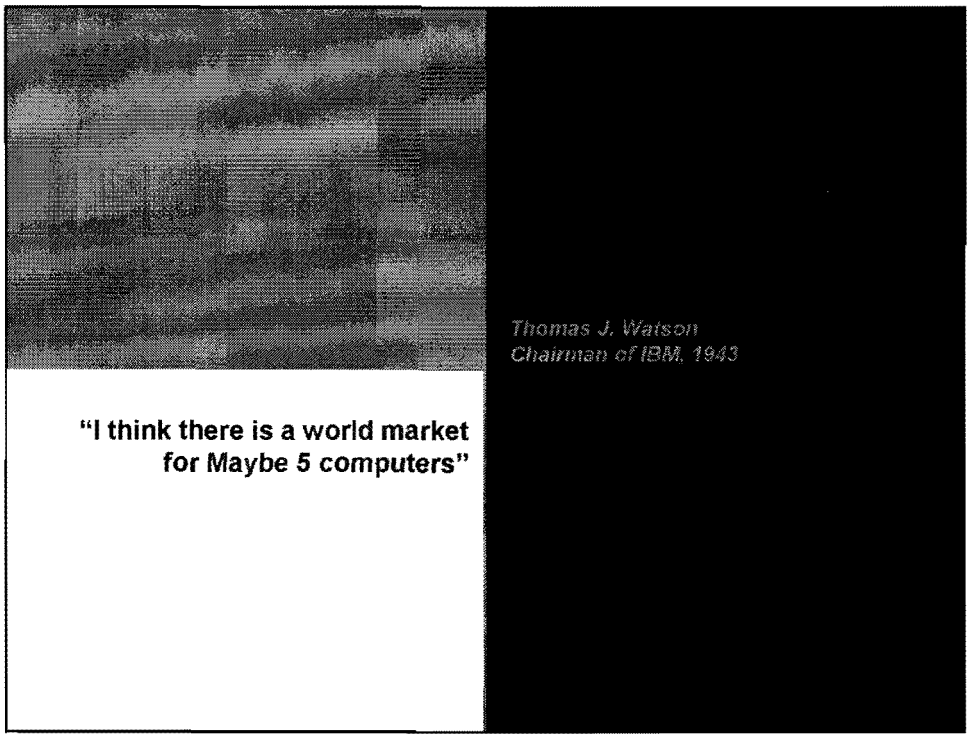
What happened ?





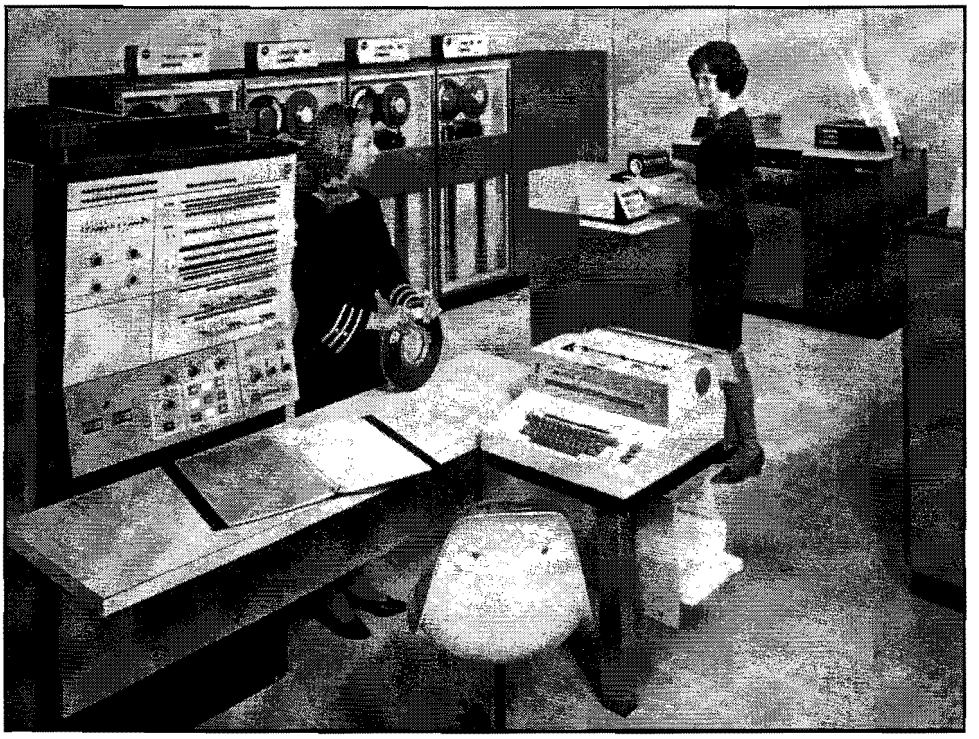
"Computers in the future may weight no more than 1.5 tons"

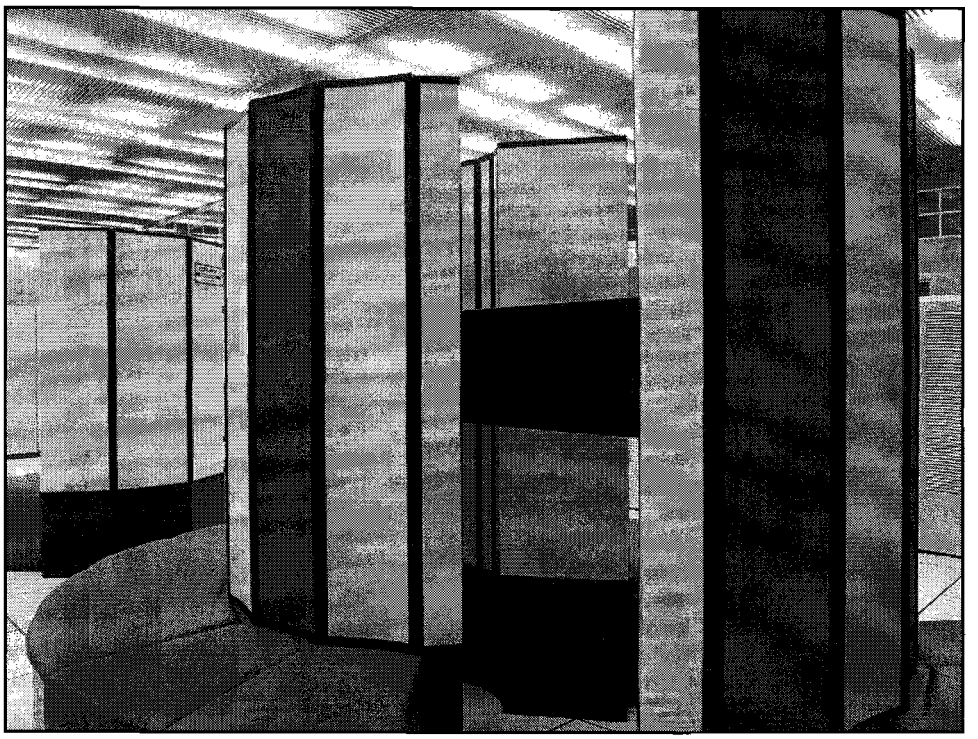
Popular Mechanics
1949

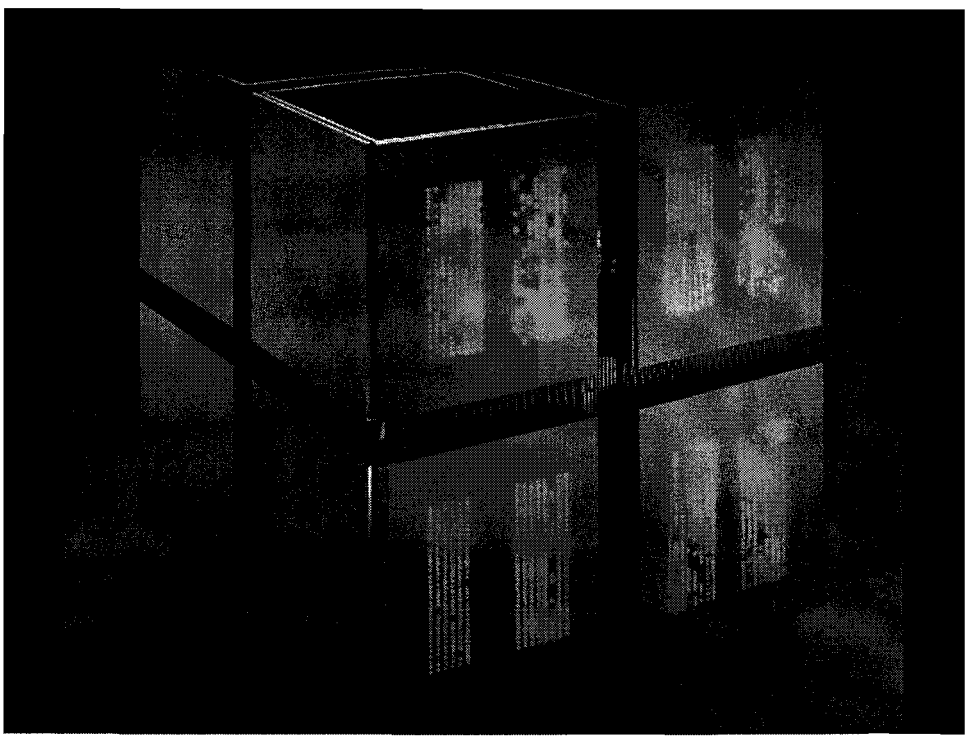


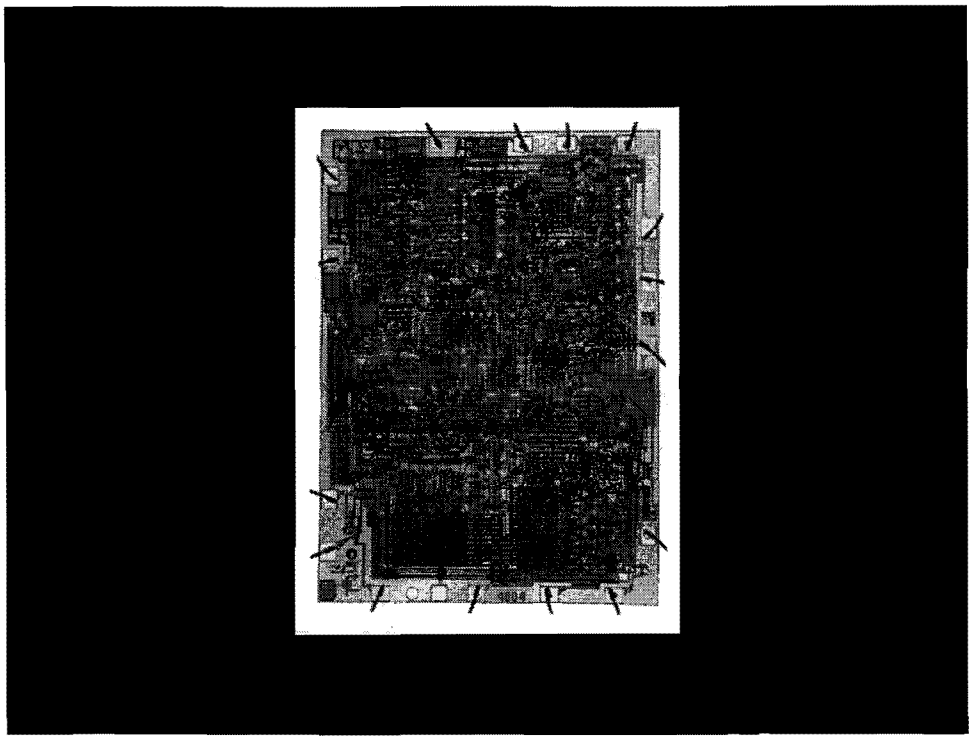
"I think there is a world market for Maybe 5 computers"

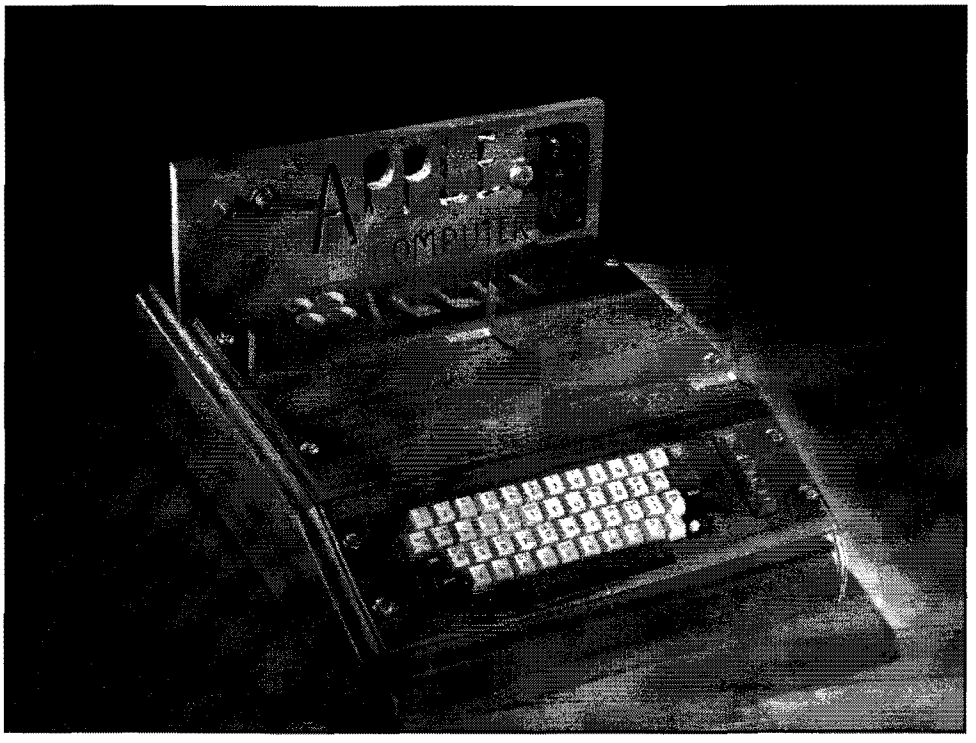
Thomas J. Watson
Chairman of IBM, 1943

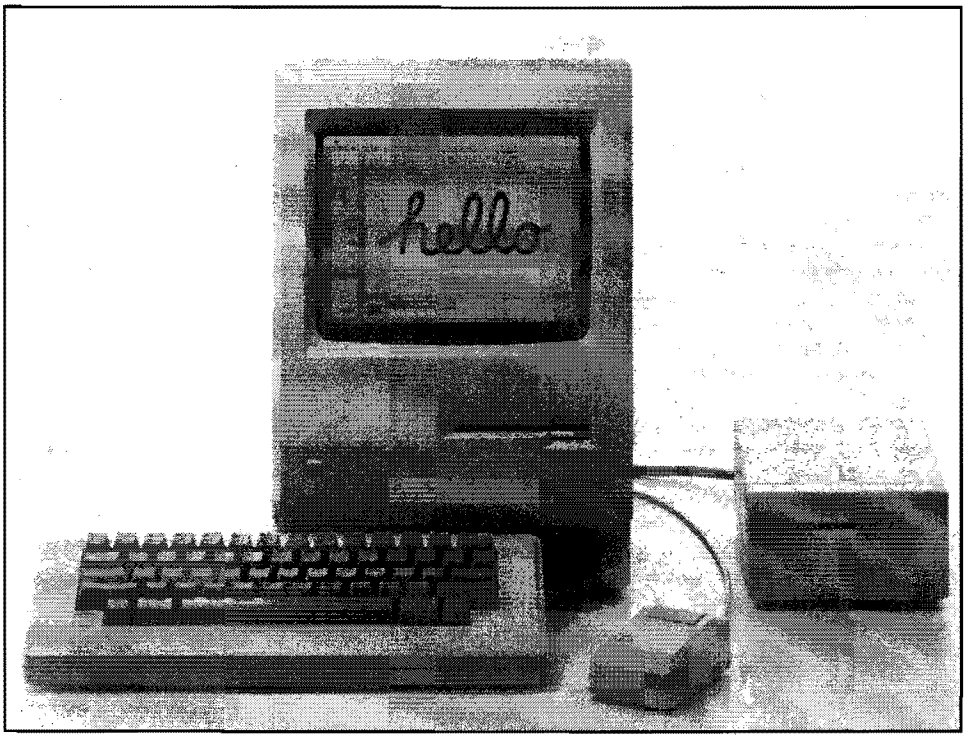


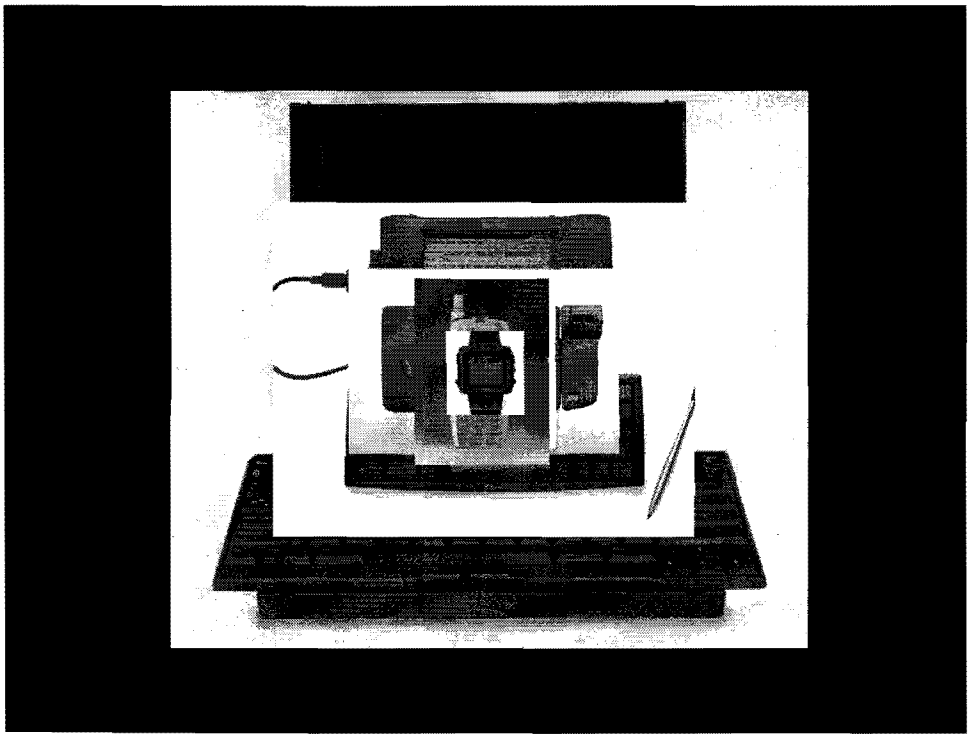


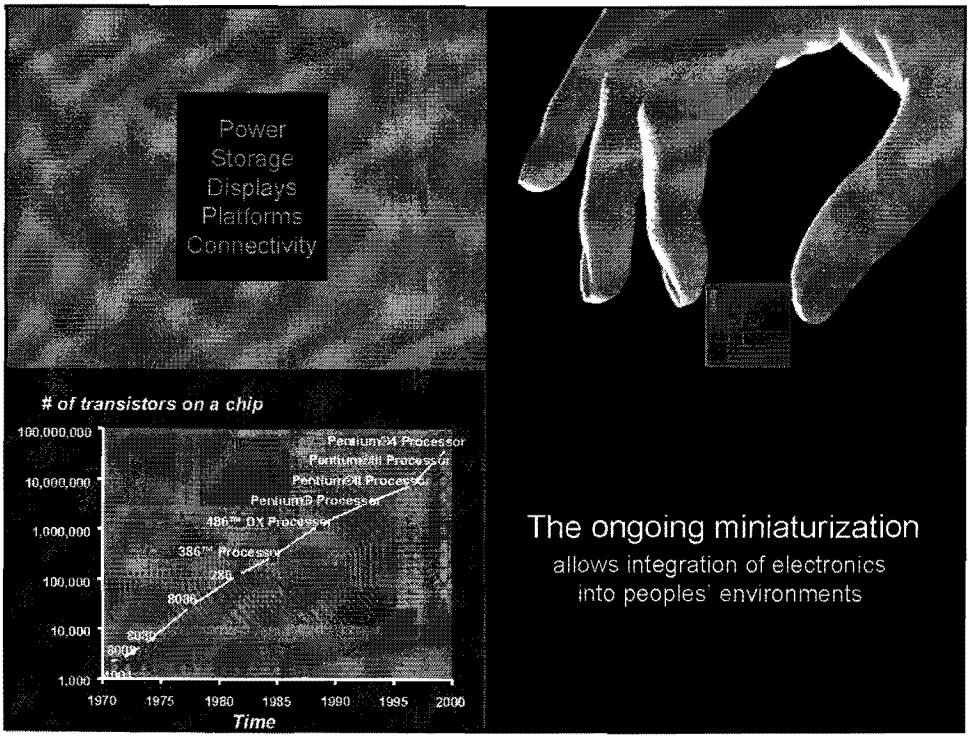


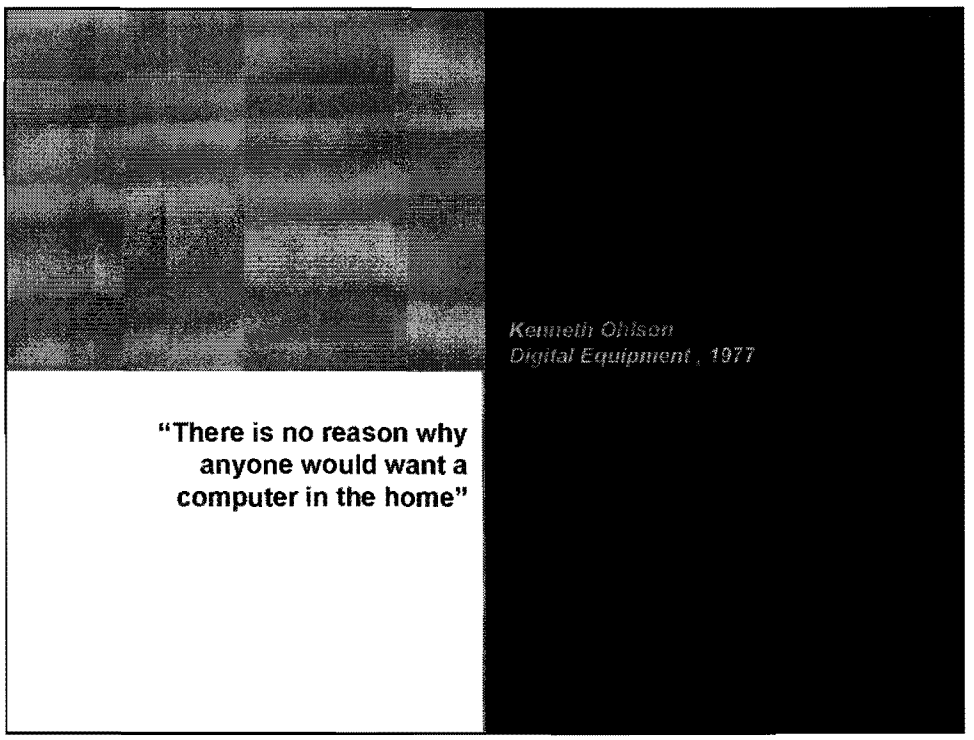


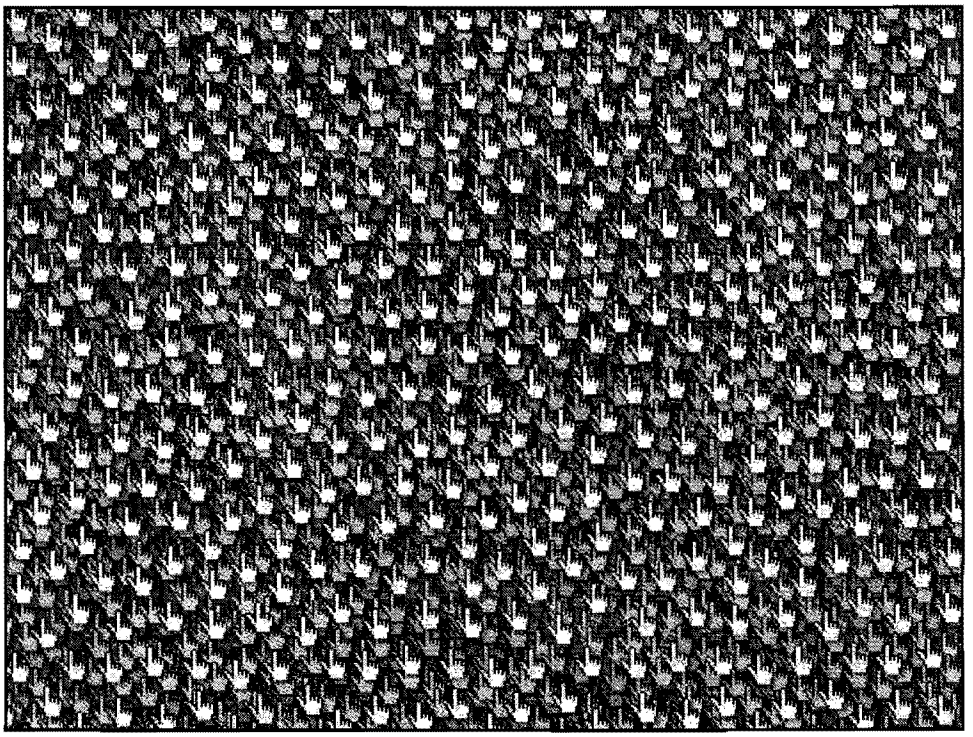


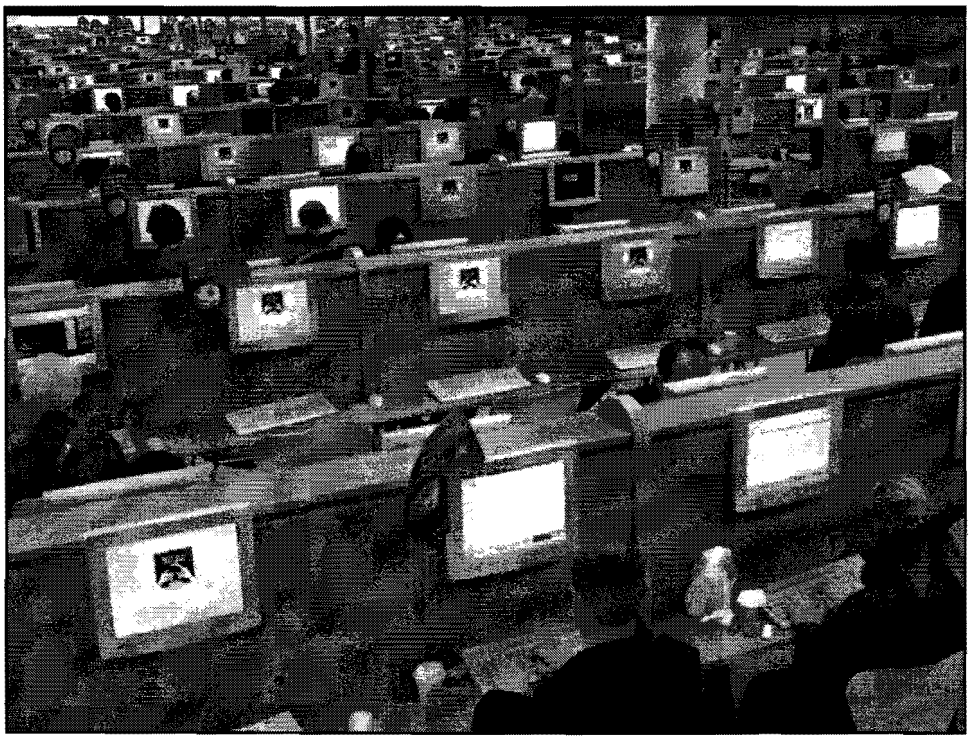

hello

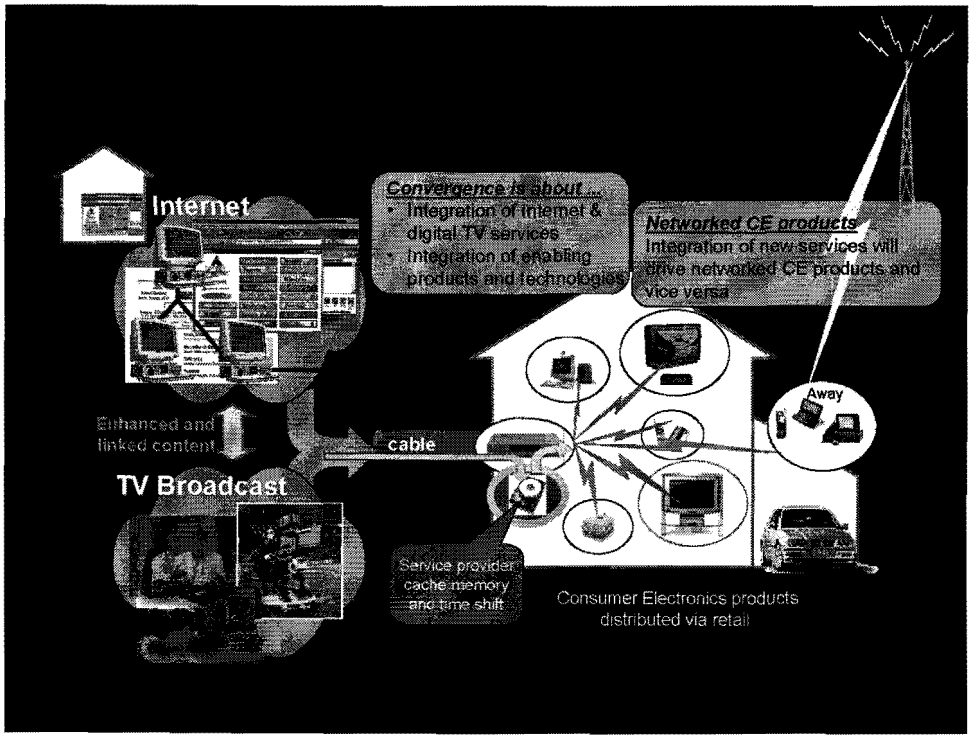



Power
Storage
Displays
Platforms
Connectivity
# of transistors on a chip
100,000,000
10,000,000
1,000,000
100,000
10,000
1,000
1970
1975
1980
1985
1990
1995
2000
Time
Pentium®4 Processor
Pentium®III Processor
Pentium®II Processor
Pentium® Processor
486™ DX Processor
386™ Processor
286
8086
8085
8080
8008
4004
The ongoing miniaturization
allows integration of electronics
into peoples' environments


"There is no reason why anyone would want a computer in the home"
Kenneth Ohlson
Digital Equipment , 1977

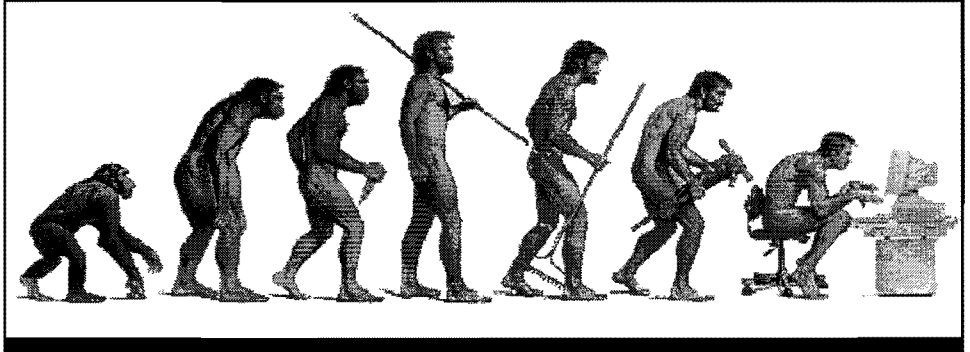


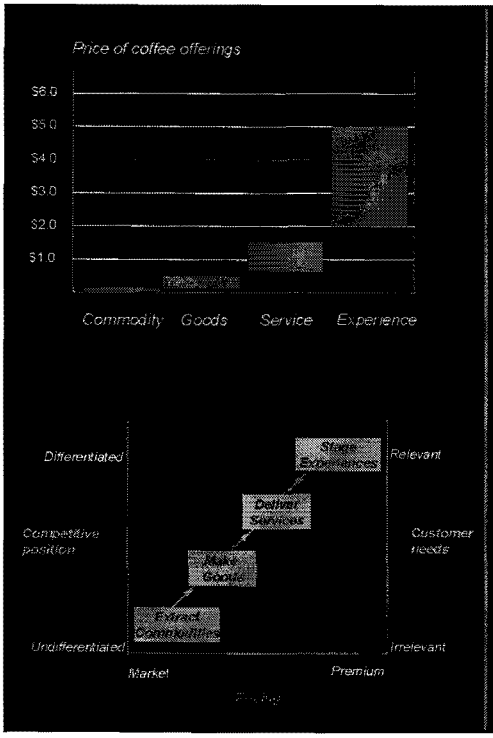


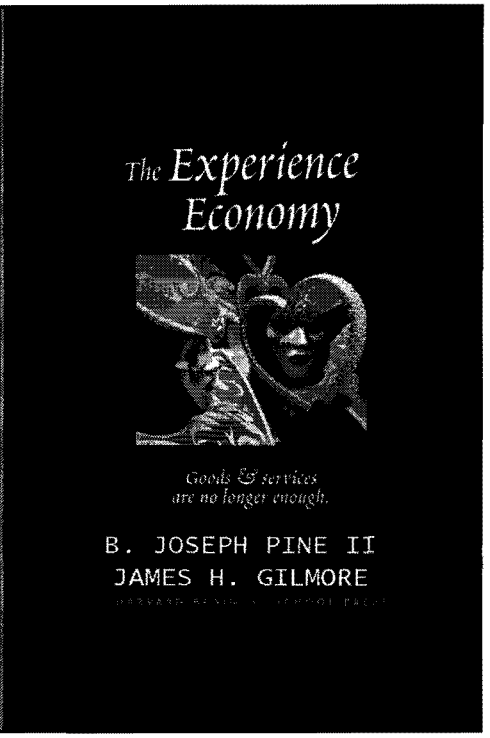

Internet
Convergence is about ...
• Integration of internet & digital TV services
• Integration of enabling products and technologies
Networked CE products
Integration of new services will drive networked CE products and vice versa
Away
Enhanced and linked content
cable
TV Broadcast
Service provider cache memory and time shift
Consumer Electronics products distributed via retail

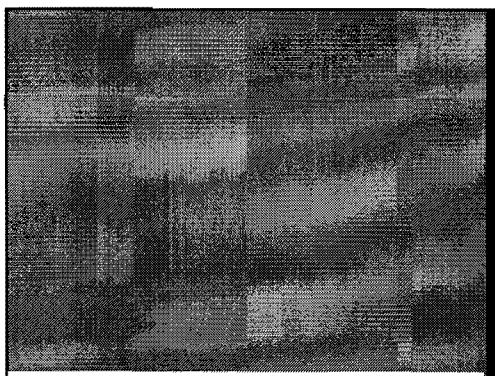
"640 Kilobyte ought to be enough for anybody"

Bill Gates
President of Microsoft, 1981




Price of coffee offerings
$6.0
$5.0
$4.0
$3.0
$2.0
$1.0
Commodity
Goods
Service
Experience
Differentiated
Relevant
Stage Experiences
Deliver Services
Competitive position
Customer needs
Make Goods
Extract Commodities
Undifferentiated
Irrelevant
Market
Premium


The Experience Economy
Goods & services are no longer enough.
B. JOSEPH PINE II
JAMES H. GILMORE

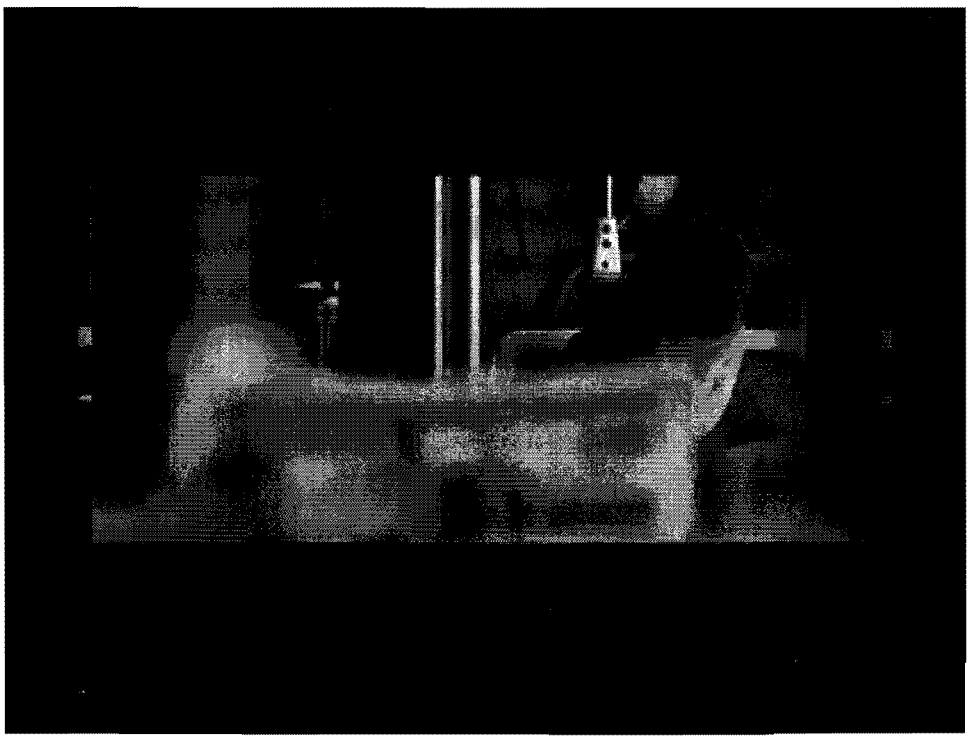


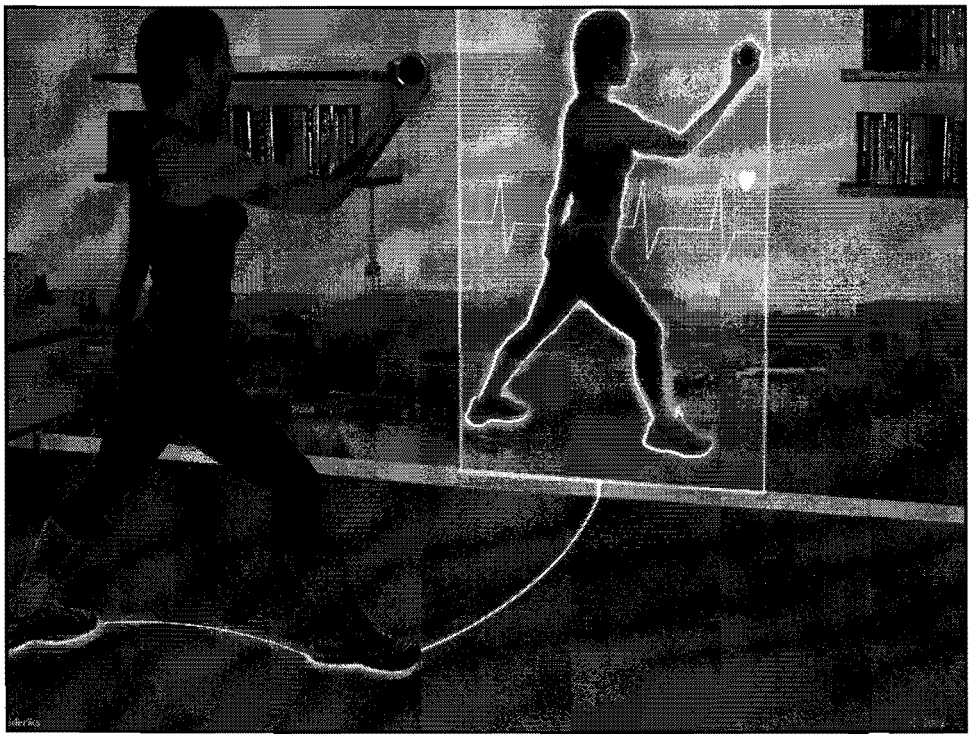
Creating the vision

Ambient Intelligence

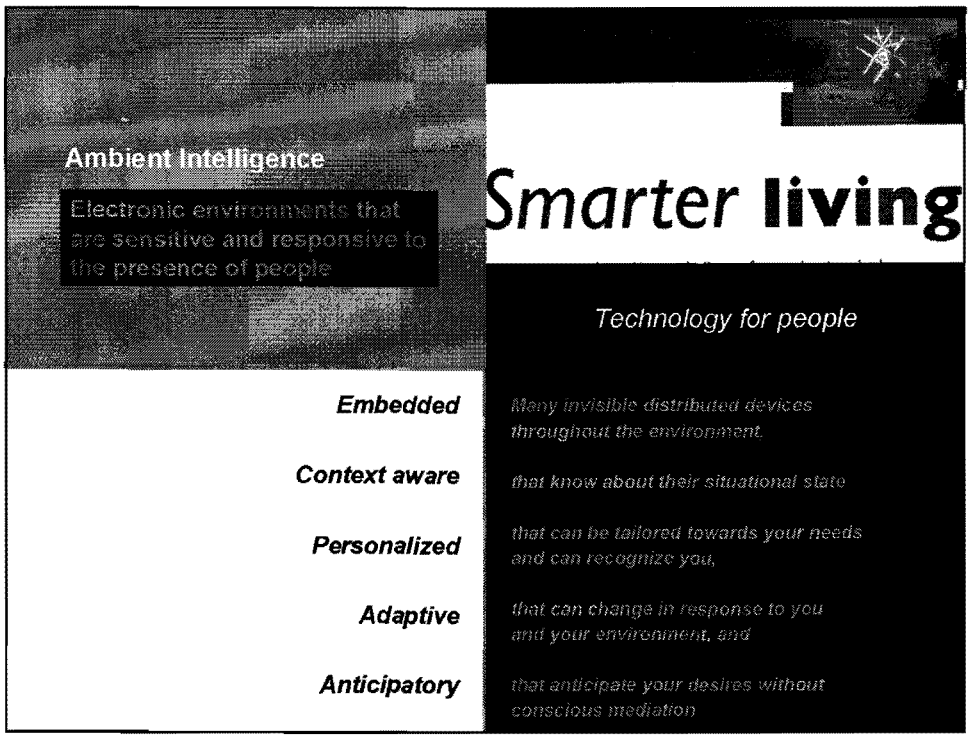
Ambient Intelligence
refers to electronic environments
that are sensitive and responsive
to the presence of people

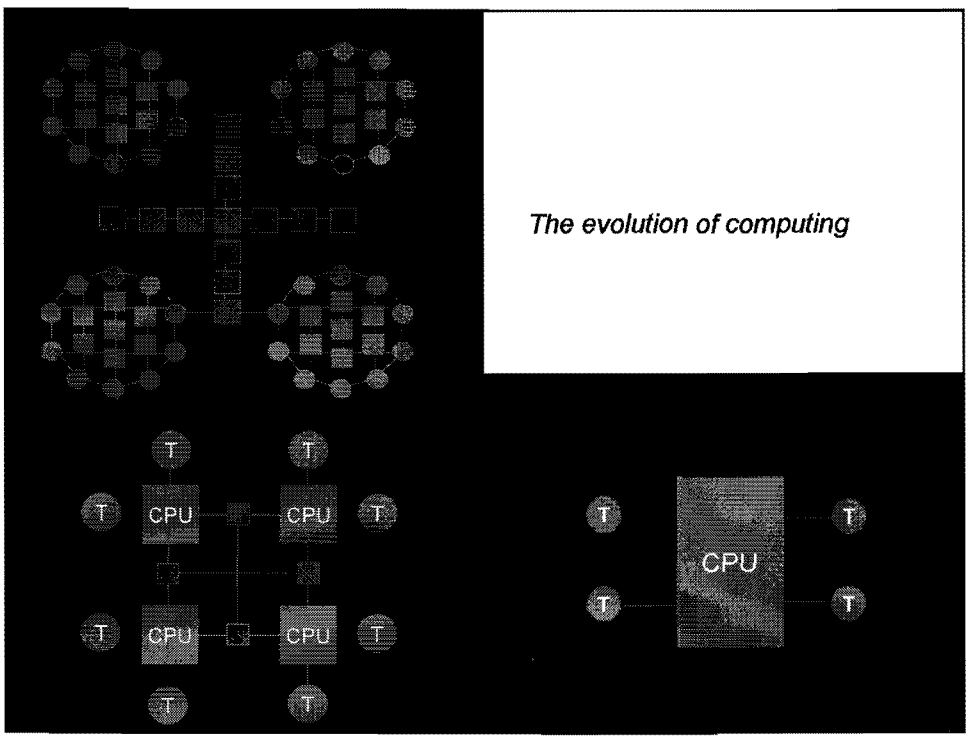
Fiction or Reality?

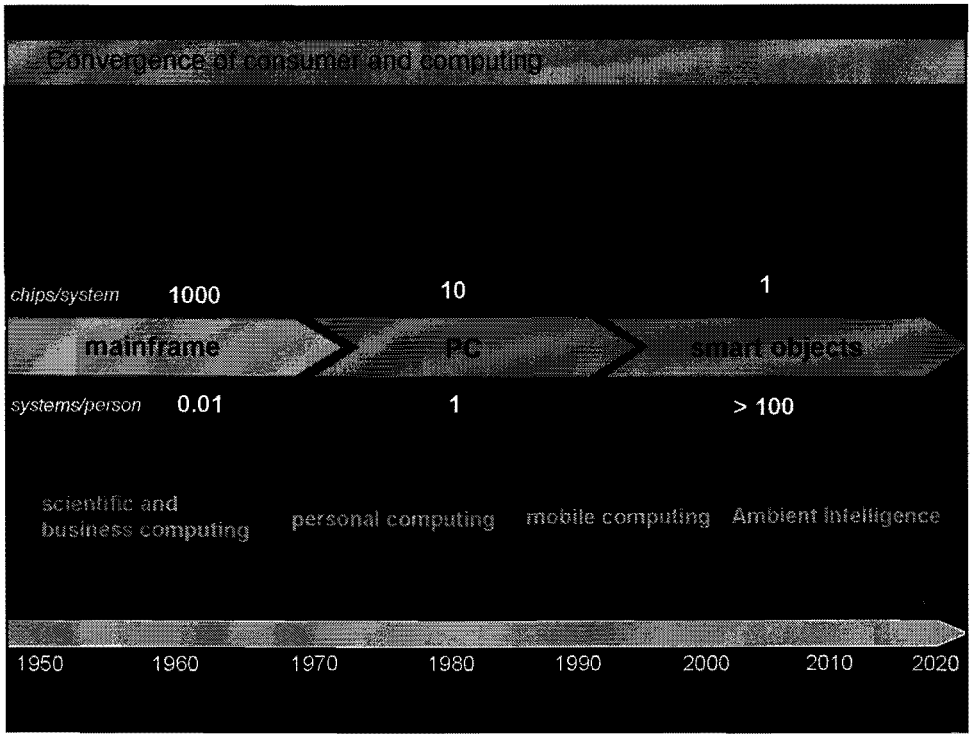


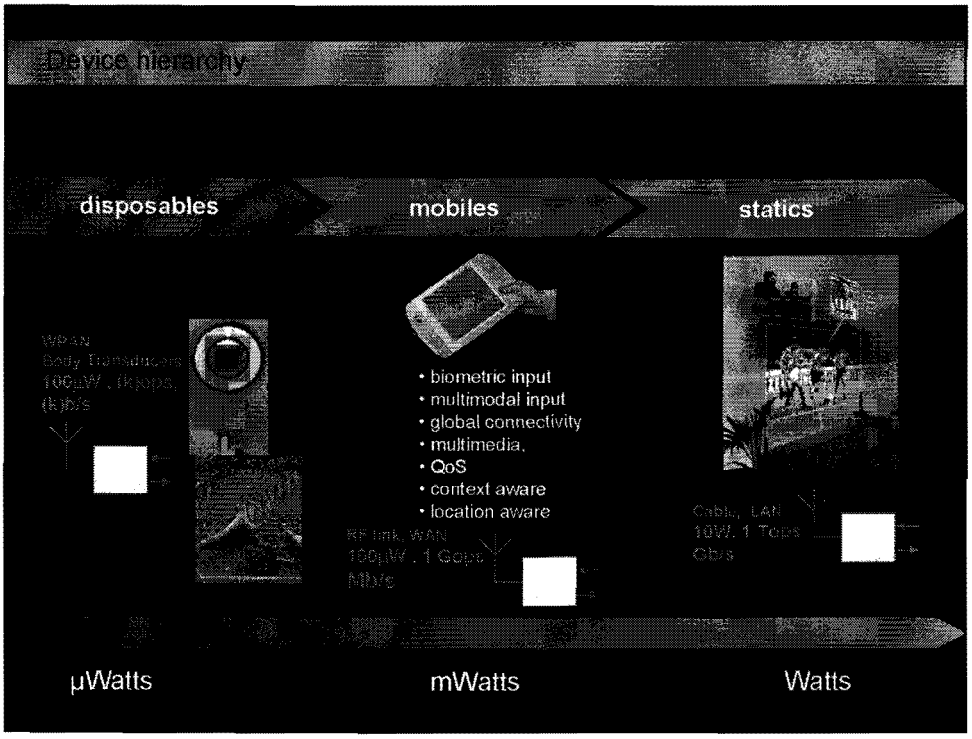



LIMP BIZKIT

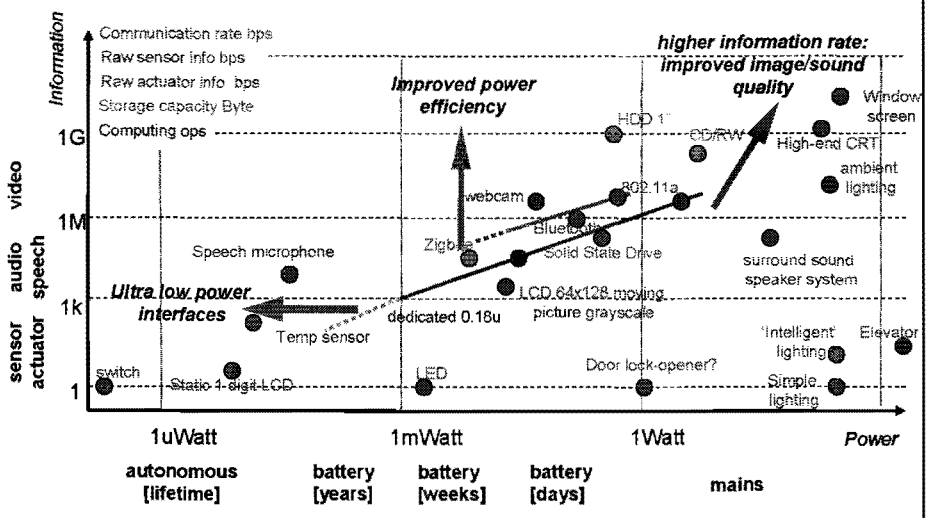

Ambient Intelligence
Electronic environments that are sensitive and responsive to the presence of people
Smarter living
Technology for people
Embedded
Many invisible distributed devices throughout the environment,
Context aware
that know about their situational state
Personalized
that can be tailored towards your needs and can recognize you,
Adaptive
that can change in response to you and your environment, and
Anticipatory
that anticipate your desires without conscious mediation

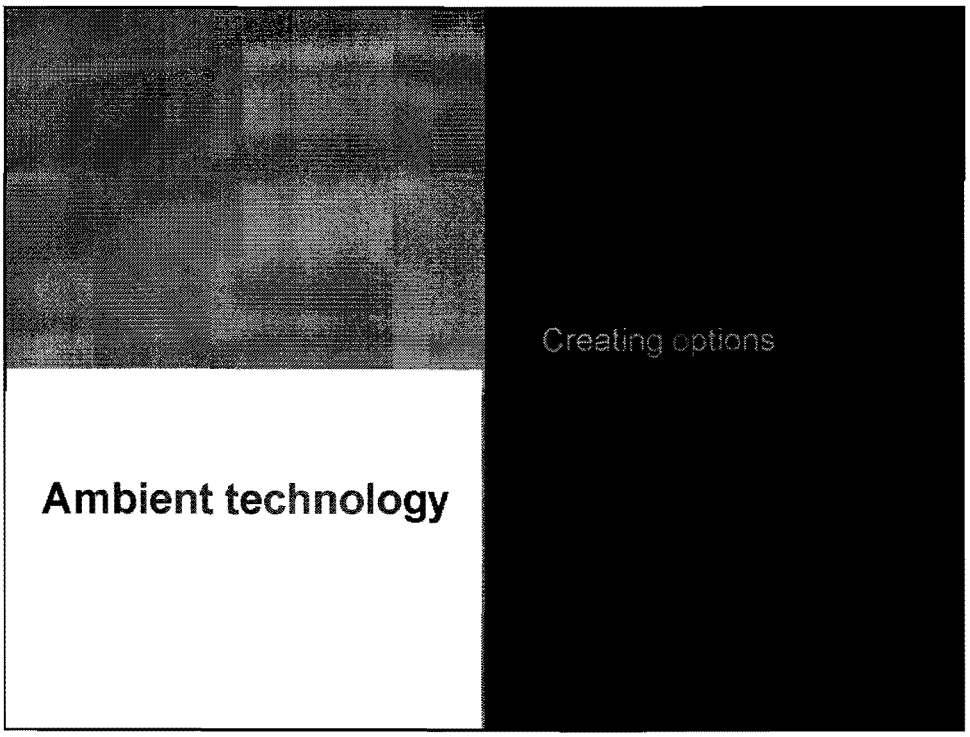

The evolution of computing
T
CPU
CPU
CPU
CPU
CPU


Convergence of consumer and computing
chips/system
1000
10
1
mainframe
PC
smart objects
systems/person
0.01
1
> 100
scientific and business computing
personal computing
mobile computing
Ambient Intelligence
1950
1960
1970
1980
1990
2000
2010
2020


Device hierarchy
disposables
mobiles
statics
• biometric input
• multimodal input
• global connectivity
• multimedia,
• QoS
• context aware
• location aware
µWatts
mWatts
Watts

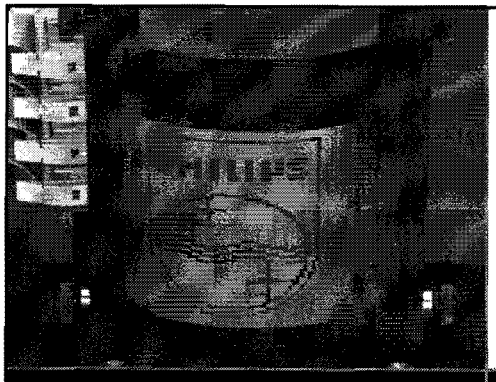
Device hierarchy





Data and content management
Computational intelligence
User centred engineering
Contextual awareness
Networked storage
Pervasive wireless
Smart materials
Electronic dust
Ambient audio
Displays
Vision
Speech
Ambient video
Ambient lighting
Trust and privacy
Internet computing
Waerable computing
Pervasive middleware
Peer-to-peer computing
Ubiquitous communication
Ambient computing platforms

Ambient technologies

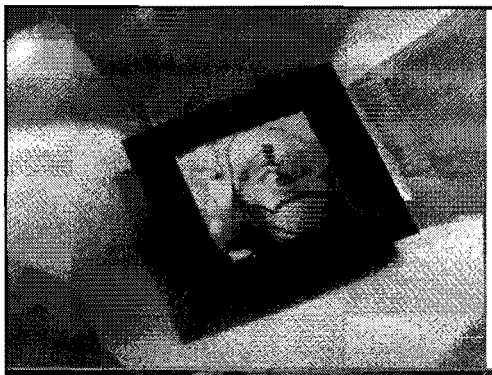


Displays

- World's first
  - working flexible matrix display
  - all-polymer active-matrix display
- Light, thin and roll-up capability
- Unlimited shape and form
- On any surface

Cholesteric texture LCD

- Passive matrix 64x64
- Thickness 250 micron
- Radius of curvature > 2 cm
- Size 12x15 cm



Displays

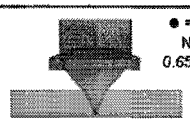
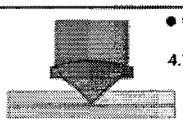
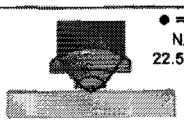
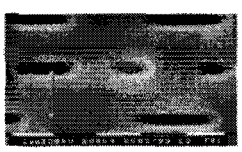
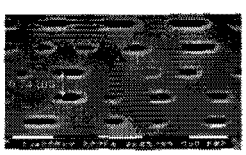
Full color inkjet-printed - polymer active matrix display

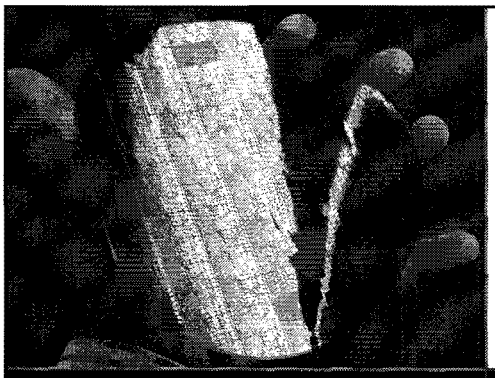
96 x (64x3) PolyLED display with 100µm x 300µm sub-pixels

Blu-ray disc for high-definition TV

- Small read-out spot achieved by
  - Blue laser
  - Very strong lens (NA = 0.85)
  - Thin cover layer
- 22.5 GB on a single side RW disc

Storage

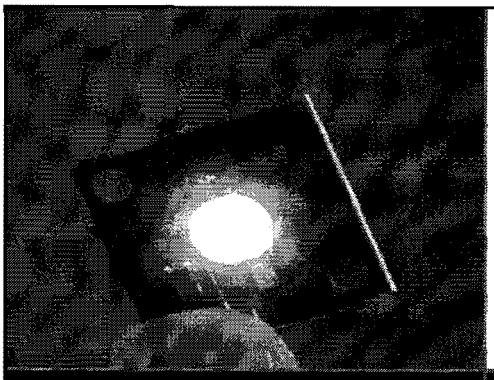
CD 0.65 GB	*DVD (2x)4.7GB*	*Blu-ray (2x)22.5 GB*
		
		



IC technology

- Standard silicon
- Full freedom of substrate choice
- Low-cost wafer-scale post processing
- Compatible with standard IC
- Assembly techniques

Silicon on anything



Lighting

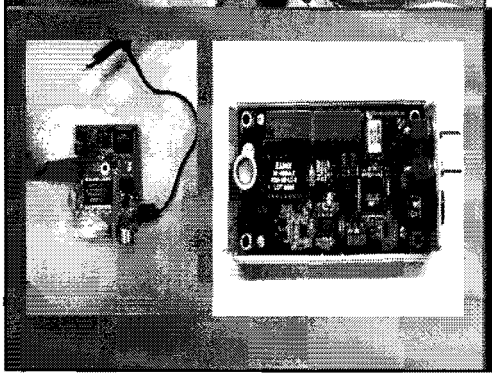
LED to Incandescent Light

- Efficacy 4 x
- Life time 100 x

Solid state lighting

Integrated light sources

Context awareness

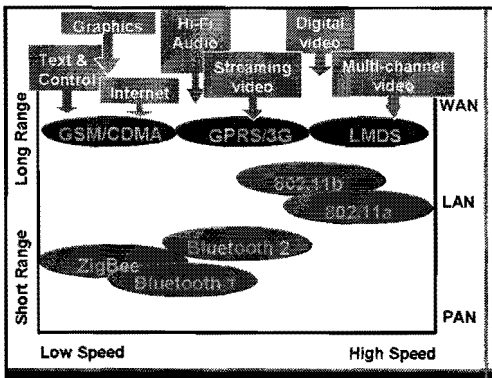


Position detection

- Beacons fixed in environment, actively sending signals
- Listeners: mobile devices that listen to beacons, passive
- Distance measurement by acoustic time-of-flight

Some examples

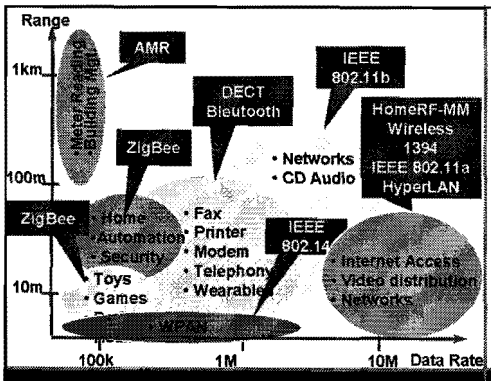
Challenges for system design



Network design

Large variety of wireless standards

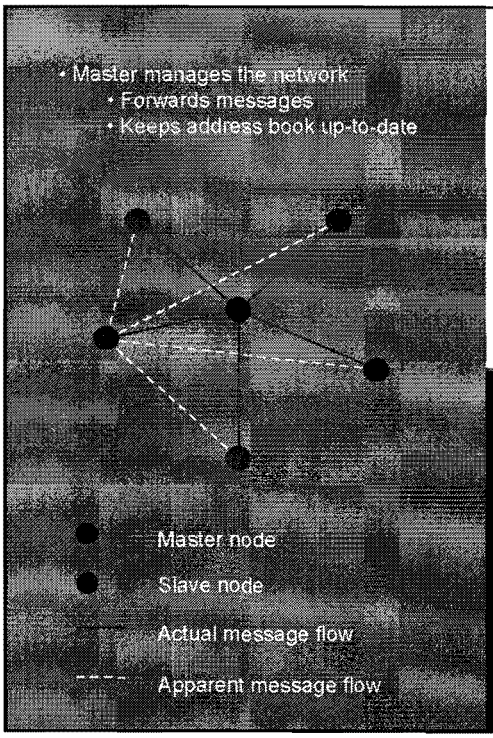
Global convergence



Network design

All ranges are covered

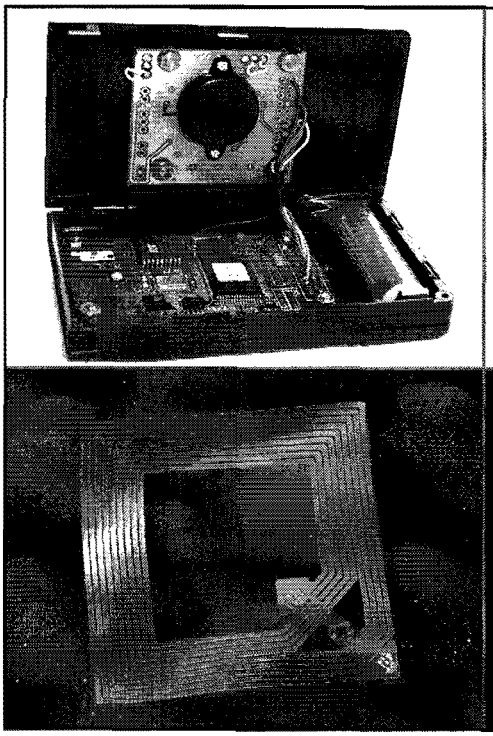
Towards ubiquitous connectivity



ZigBee networks

Tagging devices

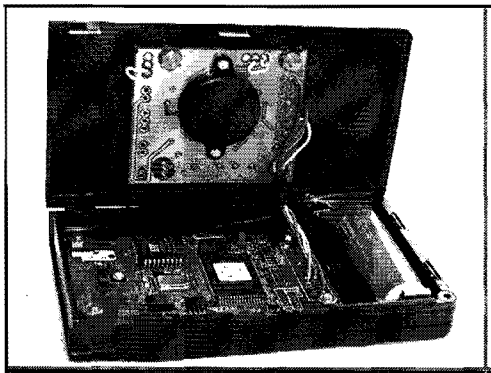
Active and passive
pico radio [100µWatt]

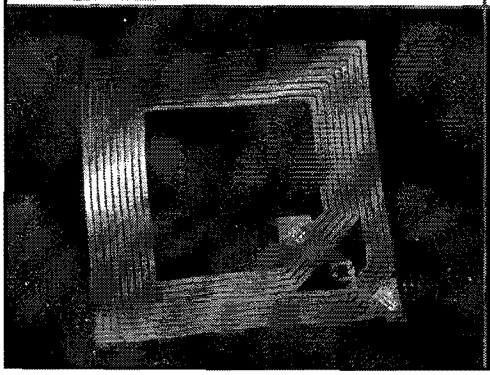


ZigBee networks

Tagging devices

Active and passive
pico radio [100µWatt]





ZigBee networks

RF data links aimed at consumer products in domestic, office, automotive, and public environments

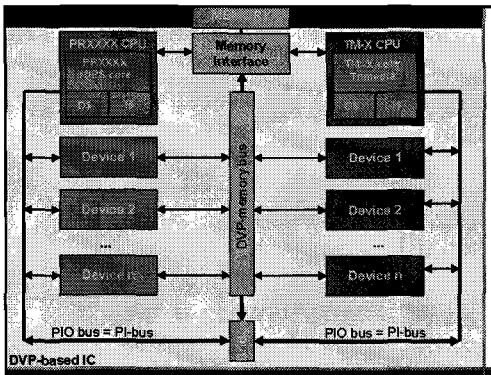
- Low cost ($1.5-$2.5)
- Low bit rate (250kb/s)
- Low power (< mWatts)
- Short range (10-75m)

- Small size (~ few mm²)
- Low power design ~ 100µW
- Integrated antenna + matching circuits
- No external frequency reference
- No external 'glue' components
- Small / integrated battery
- Power optimized protocol
- 10kbps - 115.2kbps data throughput
- 10-75m coverage range (home/garden)
- Support for up to 255 nodes
- Support for up to 100 co-located networks
- Support for 4 time-critical devices
- 0.5-2 year battery life (2 x AA)
- Up to 5ms^{-1} permitted mobility

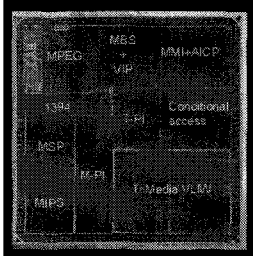
ZigBee networks

RF data link aimed at consumer products in and around the home as well as industrial sensor applications

- Low cost ($1.5-$2.5)
- Low bit rate (250kb/s)
- Low power (< mWatts)
- Short range (10-75m)



Processor design



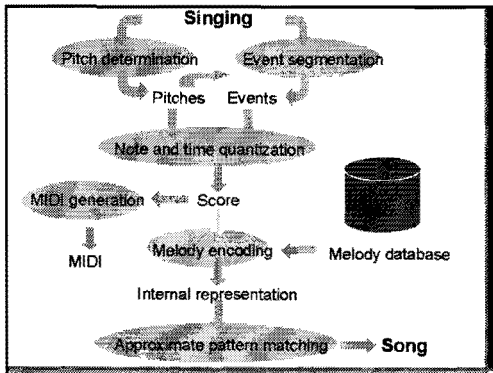
- ASB and DTV
- 0.18 µm / 8M
- 1.8V / 4.5 W
- 35 M transistors
- 75 clock domains

DVP Nexperia platform

- Domain specific
- Programmable
- Fully integrated

Processor design

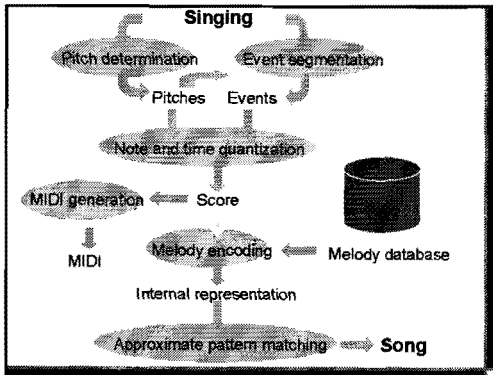
Is there a need for advanced platforms?



Computational intelligence

Intentional music search
Query by Humming

Find music titles by natural dialogue and humming



Computational intelligence

Processing steps are cascaded

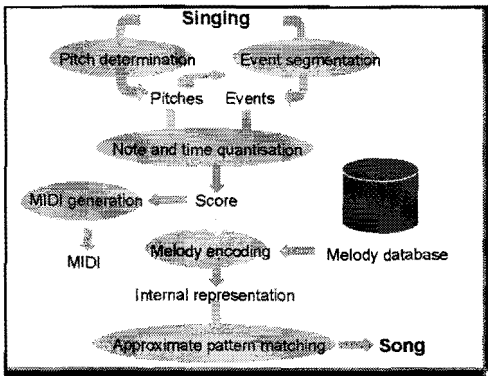
- Pitch extraction, event detection, note quantization, melody encoding, approximate pattern matching

Complexity mainly determined by approximate pattern matching

- O (n.m) where
  - n = number of notes sung
  - m= number of notes of the melody in the database

Intentional music search
Query by Humming

Computational load



Computational intelligence

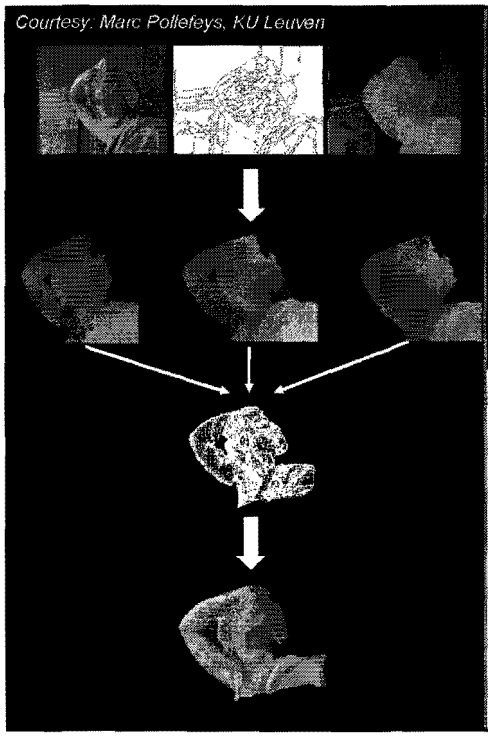
Realistic estimate

n=12, m=300 - 20000 songs, T<0.5 second

Approximate pattern matching takes 5 GOPS

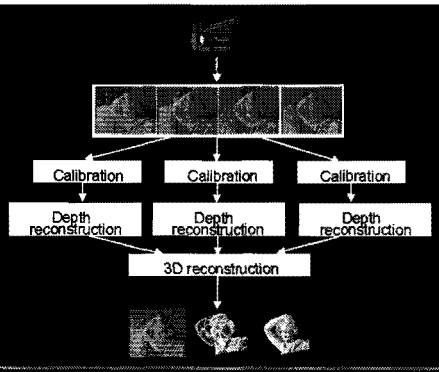
Intentional music search
Query by Humming

Computational load



Video processing

3D video construction



Video processing

Realistic estimate

- Segment-based motion estimation [5 GOPS]
- Rendering:
  - Multi-view auto-stereoscopic display, allowing multiple [720*576] images
  - Nearest neighbor interpolation: 4 GOPS
  - Bi-lineair interpolation: 20 GOPS
  - Window screen resolution 100 GOPS

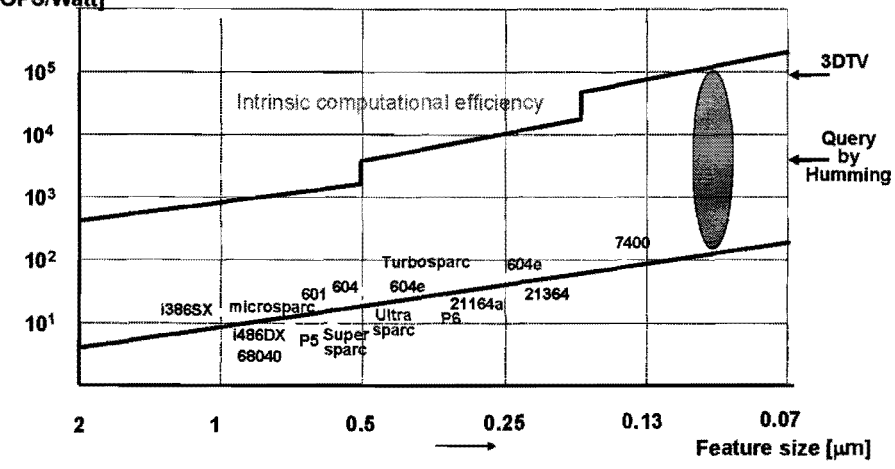
3D video reconstruction

Computational load

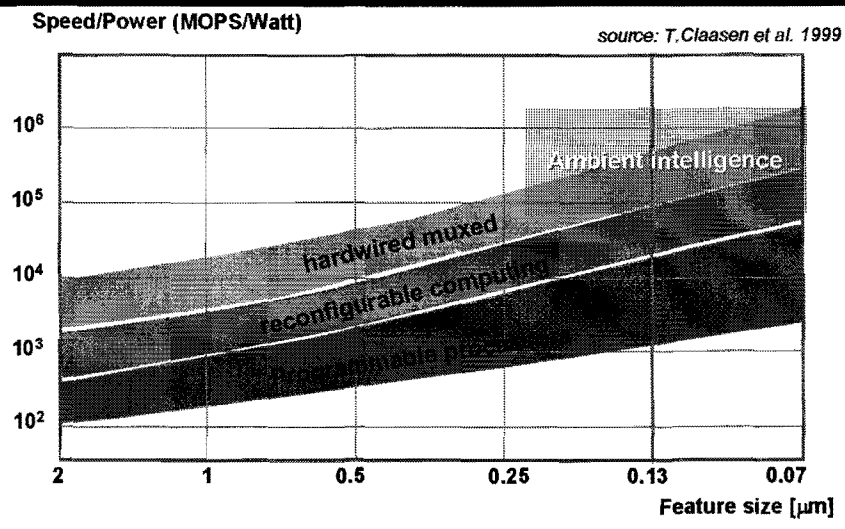
Device hierarchy

Computational efficiency [MOPS/Watt]

Source: E. Roza, 1999



Device hierarchy



ASICs

Weakly-programmable (parametrizable) functions e.g., video functions

Application specific and programmable cores (ASIP) e.g., Tensilica, AiRT VLIW

Programmable DSP cores: e.g., R.E.A.L., Palm, Oak

Programmable general purpose CPU cores e.g., ARM, MIPS, TM

+

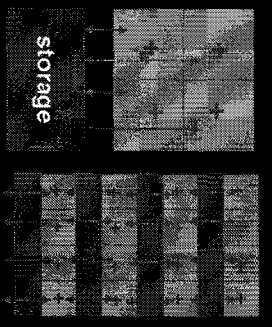
embedded memories

=

heterogeneous multi-processor architectures

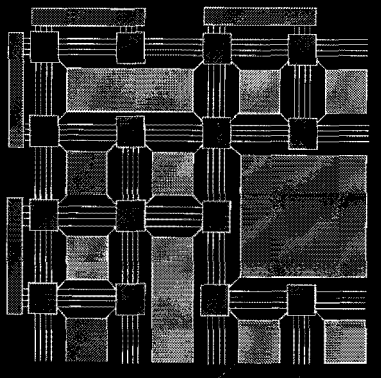
Processor design

Spatial localization
of data and
compute power



Task/Data parallelism
multi-processor network architectures

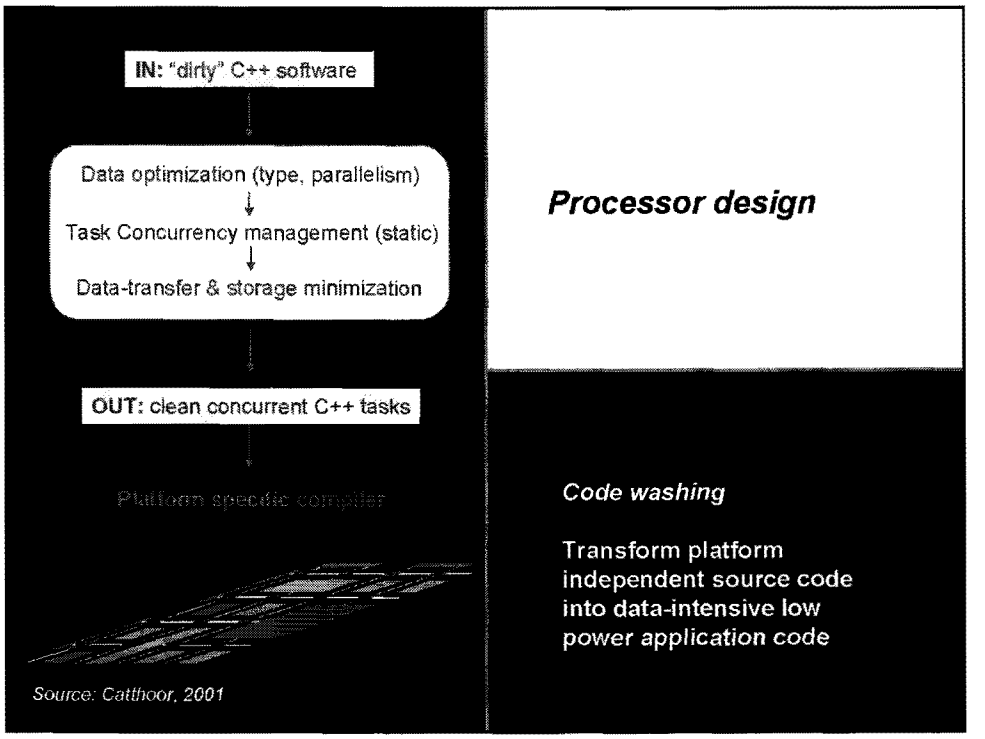
Processor design

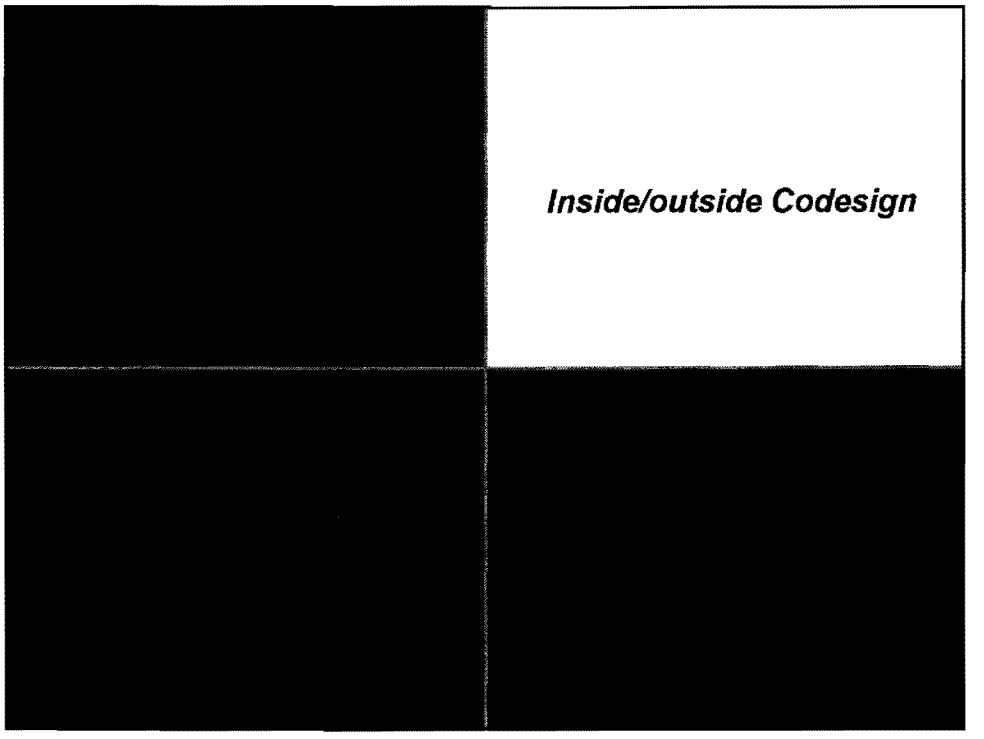


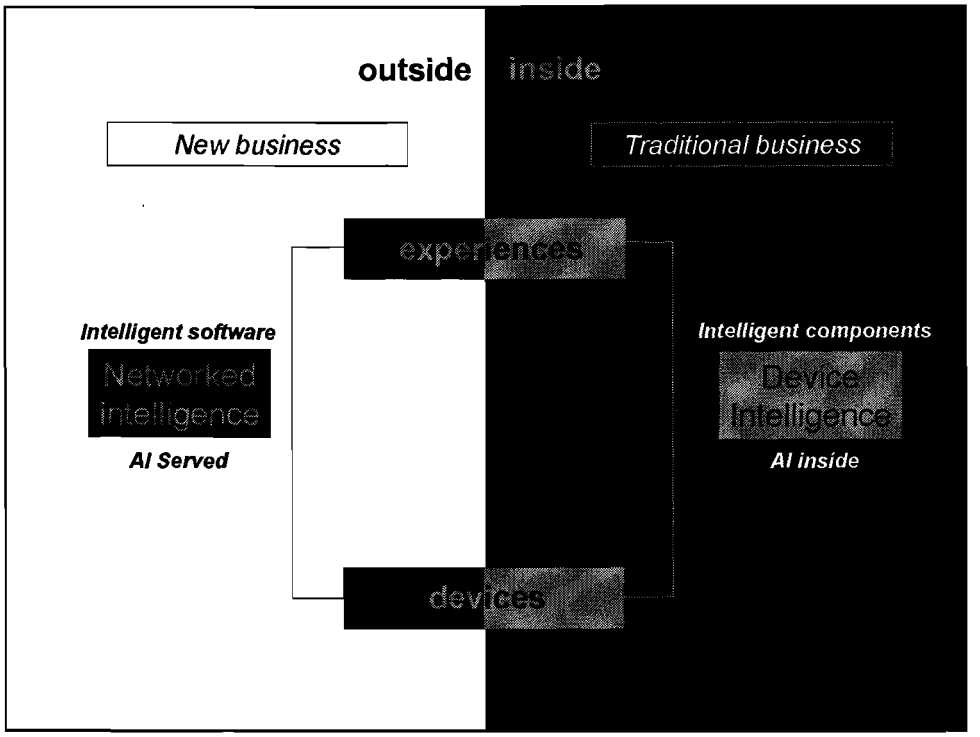
Source: Van Meerbergen, 2001

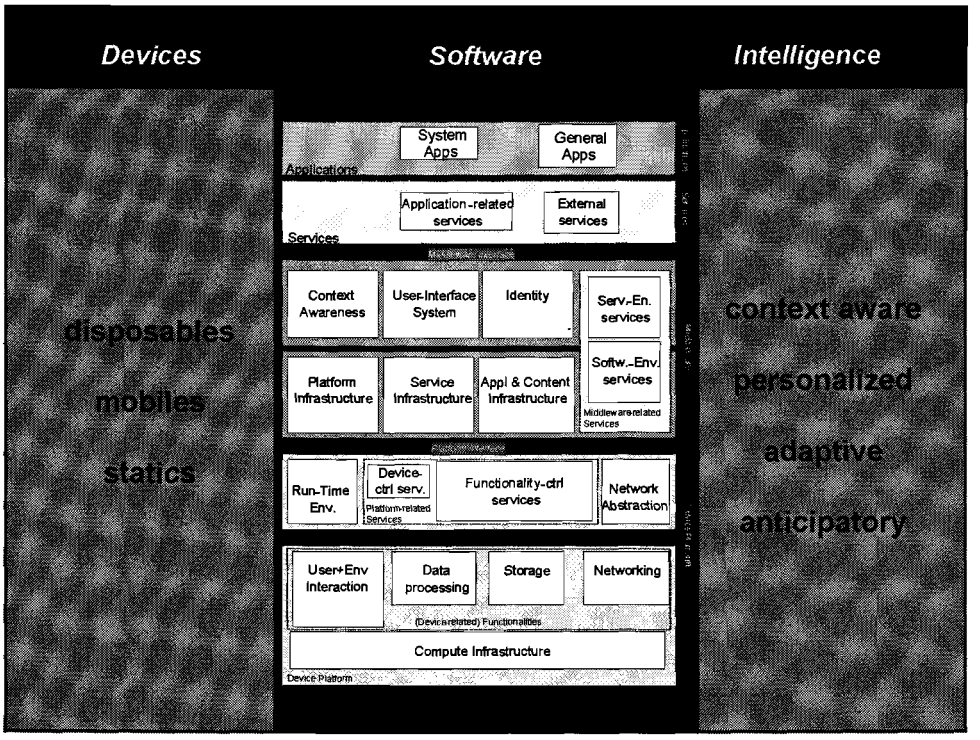
Processor design

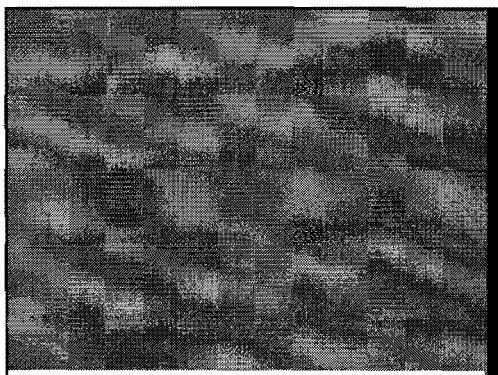
- Transistor tiles (ASIPs, memories)
- Low tile frequency (O(500 MHz))
- Locally synchronous, globally asynchronous
- RTOS controls power for each tile (V_{DD}, F_{cl}, IDDQ)
- Low data-rate communication over packet switched network
- Low power code transformation


IN: "dirty" C++ software
Data optimization (type, parallelism)
Task Concurrency management (static)
Data-transfer & storage minimization
OUT: clean concurrent C++ tasks
Platform specific compiler
Source: Catthoor, 2001
Processor design
Code washing
Transform platform independent source code into data-intensive low power application code


Inside/outside Codesign


outside
inside
New business
Traditional business
experiences
Intelligent software
Networked intelligence
AI Served
Intelligent components
Device Intelligence
AI inside
devices


Devices
Software
Intelligence
disposables
mobiles
statics
System Apps
General Apps
Applications
Application-related services
External services
Services
Context Awareness
User-Interface System
Identity
Serv.-En. services
Softw.-Env. services
Platform Infrastructure
Service Infrastructure
Appl & Content Infrastructure
Middleware-related Services
Run-Time Env.
Device-ctrl serv.
Platform-related Services
Functionality-ctrl services
Network Abstraction
User+Env Interaction
Data processing
Storage
Networking
(Device-related) Functionalities
Compute Infrastructure
Device Platform
context aware
personalized
adaptive
anticipatory

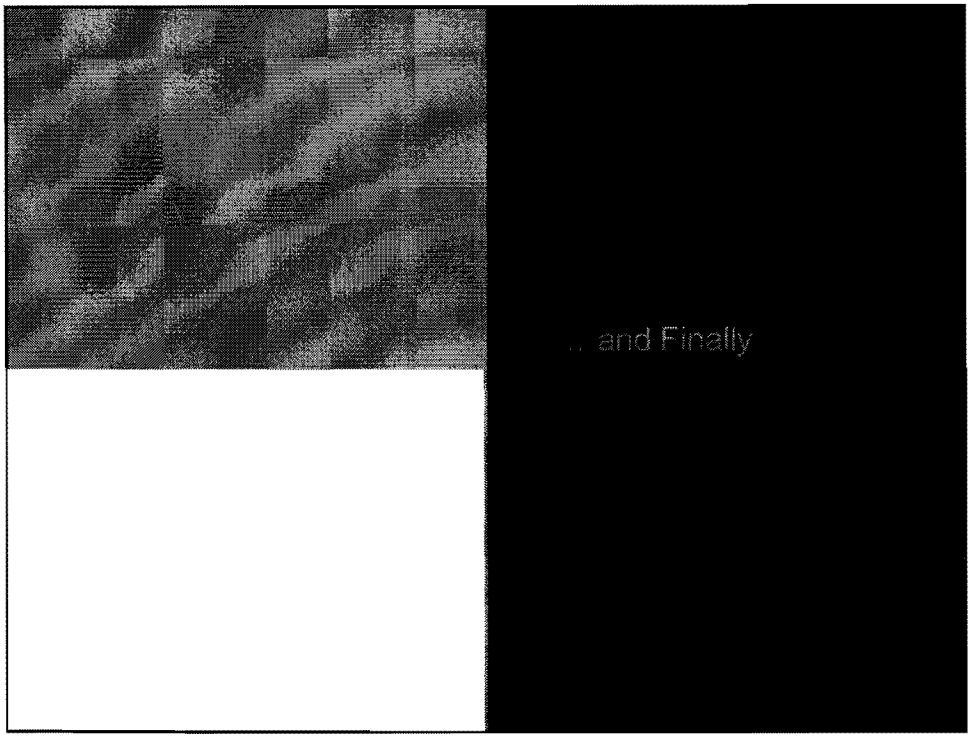


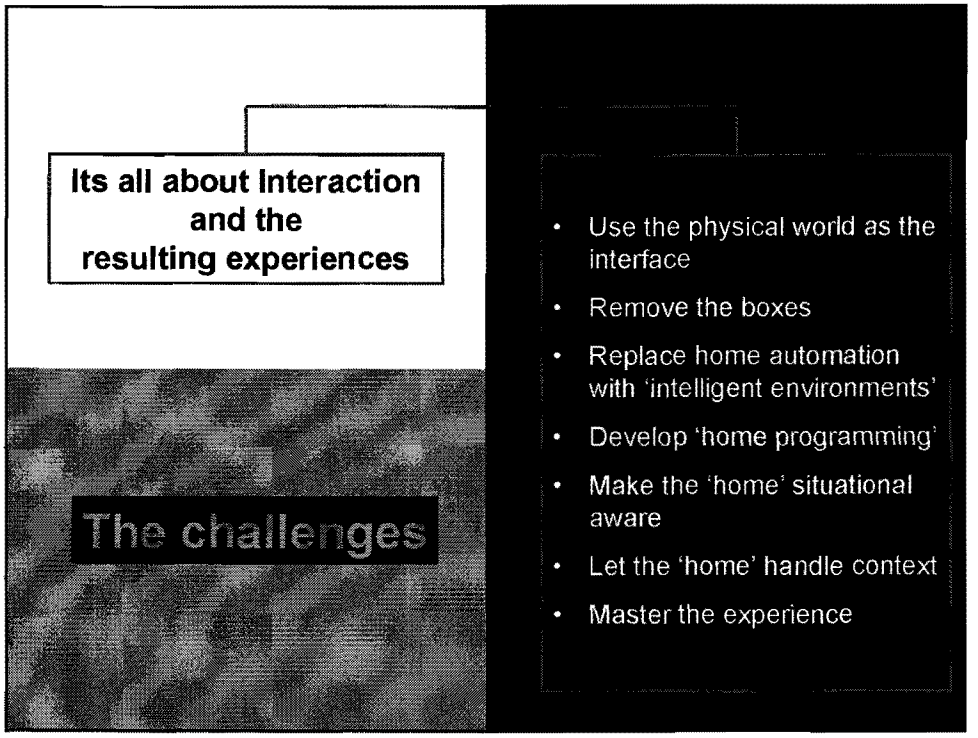
Challenges for Design

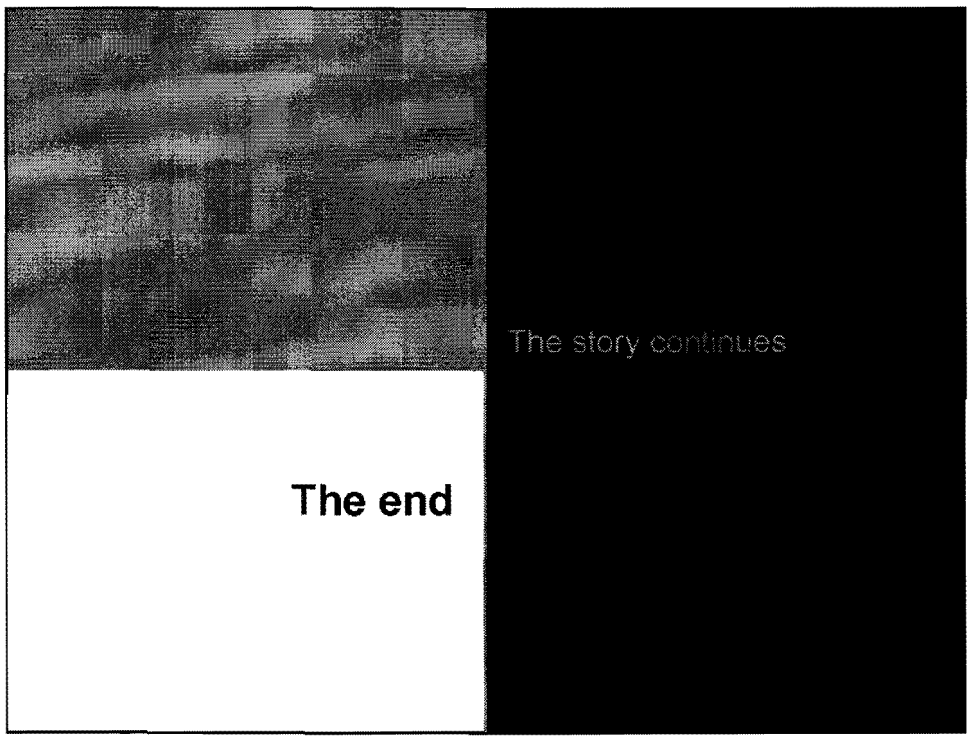
Conclusion

Design challenges from ambient intelligence

1. **A multitude of system interfaces** [1 bit/s to 5Gbit/s]
2. **Large variation in computational complexity** [1 KOPS (control) to 1 TOPS (3D video)]
3. **Large variation in power requirements** [<100 uW (no battery) to > 1W (mains)]
4. **Ubiquity presumes ultra low cost** [< 10 ct (tagging)]
5. **Design time & effort must be limited** [< 1 month]
6. **Hybrid system design** [integration of analogue, digital, RF, etc]
7. **Modular multi-processor architectures** [multi-processor networks]
8. **Multi-level system management** [application, resource, technology]
9. **Inside/outside co-design** [distributed intelligence]
10. **Multi-disciplinary design teams** [experience prototyping]


... and Finally


Its all about Interaction and the resulting experiences
The challenges
• Use the physical world as the interface
• Remove the boxes
• Replace home automation with 'intelligent environments'
• Develop 'home programming'
• Make the 'home' situational aware
• Let the 'home' handle context
• Master the experience


The story continues
The end

Scalable video algorithms for resource constrained platforms

Christian Hentschel

University of Technology Cottbus, Germany

Video signal processing is shifting from dedicated hardware to software implementation due to its flexibility. Digital signal processors (DSPs) for media processing are limited in its resources to enable cost efficient implementations for consumer devices. One way to achieve cost-efficient implementations is to use resource-quality scalable video algorithms (SVAs). This implies that dynamic resource adaptations result in dynamic quality changes which might affect the overall image quality. Starting from properties of SVAs, typical issues on quality including proposals for high-quality image processing will be presented.

SCALABLE VIDEO ALGORITHMS FOR RESOURCE CONSTRAINED PLATFORMS

Christian Hentschel

Media Technology, Brandenburg University of Technology Cottbus
Konrad-Wachsmann-Allee 1, 03046 Cottbus, Germany
phone: +49 355 692128, fax: +49 355 692150, email: christian.hentschel@tu-cottbus.de
web: www.tu-cottbus.de/mt/

ABSTRACT

Video signal processing is shifting from dedicated hardware to software implementation due to its flexibility. Digital signal processors (DSPs) for media processing are limited in its resources to enable cost efficient implementations for consumer devices. One way to achieve cost-efficient implementations is to use resource-quality scalable video algorithms (SVAs). This implies that dynamic resource adaptations result in dynamic quality changes which might affect the overall image quality. Starting from properties of SVAs, typical issues on quality including proposals for high-quality image processing will be presented.

1. INTRODUCTION

Algorithms for media processing are usually designed for a specific quality, and for many years implemented on dedicated hardware for their specific environment. For instance, in traditional television receivers various, specific ICs are combined to perform e.g. color decoding for NTSC or PAL systems, noise reduction, or frame rate up-conversion.

It is the trend of technology to develop more and more programmable platforms that allow media applications in software. Expected advantages are reduced time-to-market, re-use of hardware and software modules, portability, and flexibility. These are the issues that gain interest with respect to dedicated hardware.

The always limited computation capabilities are a restriction. This becomes especially a problem for the lower-cost, mass market processors with lower performance. On the software module side, current algorithms are designed for highest quality on given resources. They are not scalable and have a fixed functionality. It is simple to predict that the number of algorithms running in parallel is platform dependent and very limited.

Some internet applications use a kind of scalable algorithms to control the data rate. This is done by subsampling video data, by either deleting entire frames, lines, or pixels. Deleting information together with change of resolution is not acceptable in many application areas such as video processing in consumer television systems.

An alternative is to use resource scalable video algorithms (SVAs) which could solve a number of problems with respect to real-time processing on programmable platforms [1]. Figure 1 shows a range of a programmable product family versus algorithm requirements. Programmable platforms with different resources (figure 1a) will exist in parallel to suit different markets. Current media processing algorithms are designed for highest quality at given resources. In figure 1b, the height of the algorithms illustrates the resources needed for operation. The resource usage and output quality are usually not scalable, meaning that the number of algorithms allowed to run in parallel is platform dependent and very limited. A way of getting beyond these limitations is to design SVAs (figure 1c). These may have a kernel, which is not scalable (dark areas), and a part which is scalable to increase quality (light areas).

New quality issues occur since these scalable algorithms combined with QoS resource management may result in fluctuating quality with a low acceptance rate by the consumer. QoS in networks is already applied, and the differences to terminal QoS for SVAs are subject of the next section. Quality issues combined with resource usage are topics of the other sections.

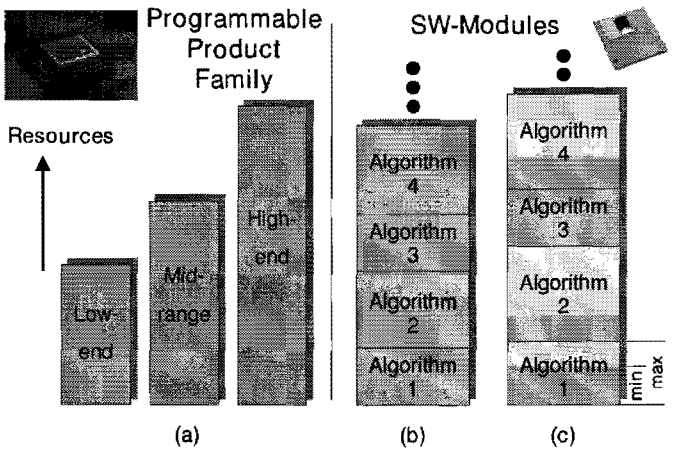


Figure 1: Programmable product family and fixed or scalable software algorithms.

2. TERMINAL QOS VERSUS NETWORK QOS

High-quality video processing in consumer terminals (CTs) has a number of distinctive characteristics when compared to mainstream multimedia processing in, for example, a (networked) workstation environment [2]. Consumer terminals need to connect to various input sources, and are increasingly being integrated in wired and wireless network environments. The transmission of various data streams including graphics, audio and video over networks started in the workstation and PC domains. From a single user point-of-view, data transmission requests are often seen as point-to-point transmissions. Network requests from other users are independent and these additional activities are recognized by the single user only because of long delays or even network access denials. The most limiting resource is network bandwidth, which has to be shared by all current users. To solve these transmission problems, QoS has been introduced to optimise the service between different users. Data streams may get priorities and a specific portion of the network bandwidth. Typical QoS parameters for streaming video over networks are image resolution, image size (window), frame rate, color depth, bit rate and compression quality in order to lower the transmission bandwidth. In summary, network QoS trades transmission bandwidth resources to optimise the overall quality.

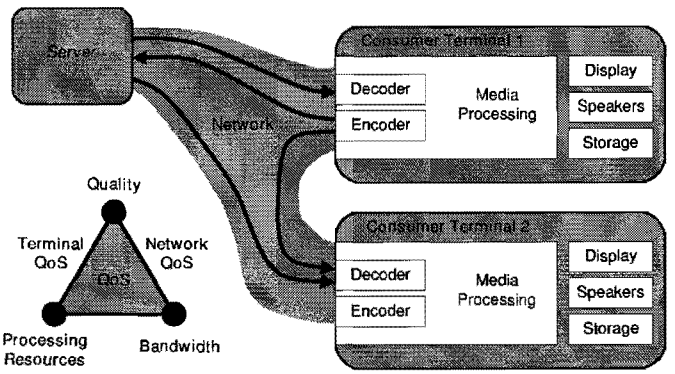


Figure 2: Differences between terminal QoS and network QoS.

Figure 2 shows an example of two future consumer terminals and a server in a network environment [3]. These terminals provide decoders and encoders, which are parts of the media processing and network interfaces. Output devices may be displays, speakers and storage devices. The different applications (e.g. view a movie, access the internet, play games, etc.) require different media processing algorithms, which also depend on the input data (resolution, frame rate, quality, etc.).

Typically, the processing resources are the limiting factor in consumer terminals, and not the bandwidth resources. Therefore, QoS in consumer terminals is different from that in networks. Multimedia con-

sumer terminals usually have a fixed resolution. The image size is determined by the display or chosen by the end-user, but not by the system. Consumer terminals have real-time requirements and do not allow frame rate fluctuations or audio interruptions.

As depicted in figure 2, terminal QoS trades processing resources over quality. The triangle also connects processing resources with bandwidth. An example for the validity of this triangle is an MPEG transmission. With the given transmission bandwidth, the data quality can also be influenced by the encoder processing resources. With more processing resources, a higher quality can be achieved at the same transmission bandwidth.

CTs such as TV sets and Set-top boxes are currently receivers in a broadcast environment, and therefore do not have the option to negotiate compression quality and bit-rate, although that may change in the future for CTs in an in-home digital network.

3. PROPERTIES OF SCALABLE VIDEO ALORITHMS

A scalable algorithm is an algorithm that:

- allows the adaptation of quality versus performance on a given architecture,
- supports different software and/or hardware platforms for media signal processing, and
- is easily controllable by a control device for several predefined settings.

A set of scalable algorithms in a modular form can perform the different applications needed in a set-top box, TV set, multimedia PC, or, more in general, media processing unit.

3.1 Basic scalable video algorithms

Figure 3 shows a block diagram of a scalable algorithm. The algorithm for media processing consists out of different functions. Some of them are scalable for several quality levels, but there is no need that all of them have to be scalable. The outcome of the scalable algorithm is dependent on the appropriate combination of the quality levels of the functions (Figure 4). These combinations may vary a lot, but only few of them may provide acceptable quality levels for the scalable algorithm.

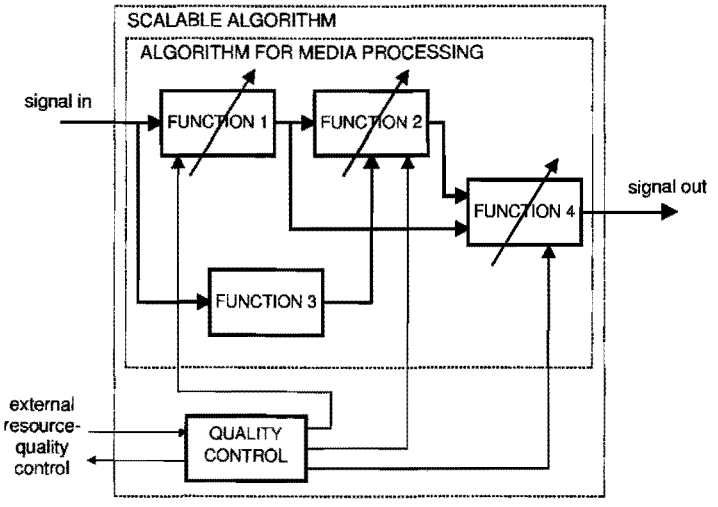


Figure 3: Basic structure of a scalable algorithm.

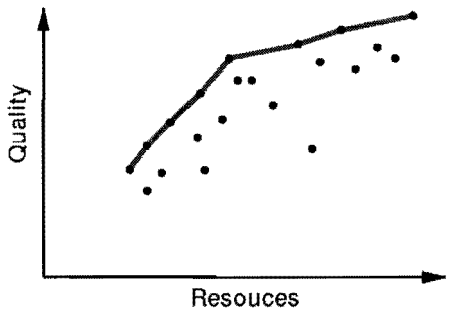


Figure 4: Best choices of quality-resource combinations for functions of the entire scalable algorithm.

The control signal for quality level can be as simple as the chosen quality level or guaranteed budget/resources (Figure 3). The block 'quality control' has specific knowledge about the algorithm for media processing and the best combinations of settings for the several functions. Because of this specific knowledge, it becomes an important part of a scalable algorithm.

3.2 Example of a simple SVA with data-independent resource usage

The block diagram of a scalable down-scaler is shown in figure 5. Picture-in-picture (PiP) applications require only simple down-scaling by natural factors, so the algorithms only includes a low-pass filter, followed by the down-sampling [3]. In this case, the down-scaling is restricted to a factor of four, and just the low-pass filter is scalable. Only output pixels need to be calculated, which reduces processing resources significantly.

The scalable down-scaler with a decimation factor of 4 in both the horizontal and vertical directions requires between 4-14 MIPS (table 1). At the lowest quality level QL0, subsampling alone without any pre-filtering is performed. Quality level QL1 uses an average filter over 4x4 pixels for the luminance, while QL3 uses the same filter for the chrominance, too. Separable 5-tap filters for the luminance (QL2) and for both luminance and chrominance (QL4) complete the scalability range (table 1). The scalability ranges from 29-100 %, corresponding to a resource range of 10 MIPS. Quality in figure 6 was coarsely estimated by a few expert viewers.

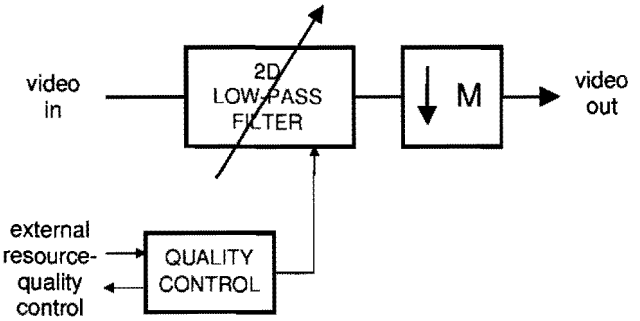


Figure 5: Basic structure of scalable image down-scaler.

Quality Level	Resource Usage [MIPS]	Low-Pass Filter Luminance	Low-Pass Filter Chrominance
0	4	off	off
1	8	average over 4x4 pixel	off
2	9	separable filters over 5x5 pixel	off
3	12	average over 4x4 pixel	average over 4x4 pixel
4	14	separable filters over 5x5 pixel	separable filters over 5x5 pixel

Table 1: Quality levels and functional details of the resource-scalable spatial down-scaler

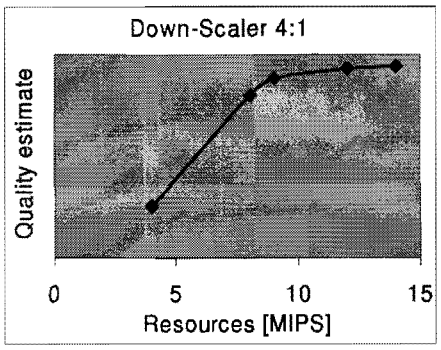


Figure 6: Best choices of quality-resource combinations for functions of the down-scalers for PiP applications.

Scalable algorithm with quality scalability in exchange to compute requirements have typically an essential core to perform its minimal function. The quality can then be increased, depending on the available computation resources. Typical quality-resource behaviour of an SVA is shown in figure 7. At minimal resources the quality can be very low, with a steep increase in quality for increasing resources. An example is a spatial scaler with low resource usage by pixel subsampling (down-scaling) or repetition (up-scaling) with poor output quality. A bilinear interpolation with few additional resources increases the quality significantly. Further quality improvements by using high-order polyphase filters become smaller compared to the resources required. As a result, a wide range of resource scalability is possible within a small range of quality. The area of high quality changes at small resource changes should be avoided for scalable media processing.

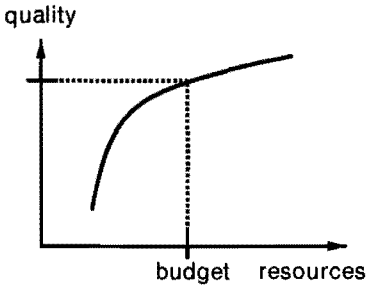


Figure 7: Typical quality-resource behaviour of scalable algorithms.

The resource-quality behaviour in figure 7 is simplified and cannot reflect all real properties. Resources are multidimensional and can include CPU cycles, memory, bus bandwidth, coprocessors, etc. Quality can be measured in a multidimensional space as well. Properties such as sharpness, color reproduction, noise, quantization, algorithm specific compression artefacts can be expressed in quality, but also more specific properties such as resolution in x- and y-direction, temporal resolution, judder, etc. A research topic is how these parameters influence each other, especially in a dynamic environment.

3.3 SVAs with data dependent resource requirements and quality

Programmable components are most suitable for irregular processing. Compression algorithms and non-linear processing algorithms such as motion estimation have such irregular processing [4], while video processing algorithms such as scaling, image enhancement etc. are mostly executed by pipelined regular pixel processing. Thus video processing on programmable components require different kinds of algorithms to ensure efficient and effective processing at available resources.

A different processing approach is illustrated in figure 8. The advantage of irregular processing is the option to do input data dependent image analysis and choose for a strategy how to process the data. An example is priority processing of sharpness enhancement in images. All relevant edges must be detected which could appear sharper by adding detail information. In flat, unstructured regions the addition of detail information would increase the noise level which lowers subjective image quality. With priority

processing, first the most relevant edges should be enhanced, followed by lower priority regions. Flat regions should not be processed, or, in case of still available resources, could be processed by noise reduction algorithms.

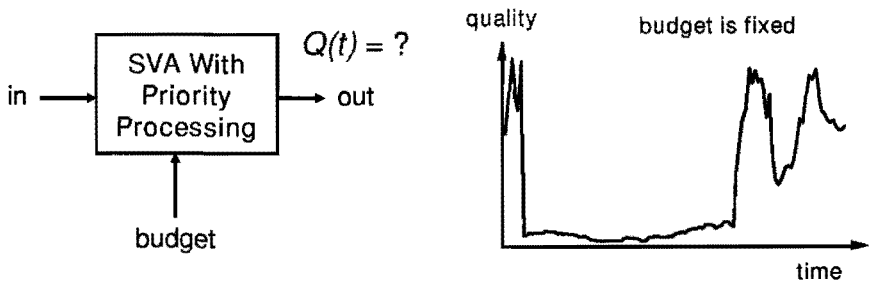


Figure 8: Fixed processing budget/resources and data dependent output quality.

An advantage of priority processing is to be able to interrupt image processing in case of low resources while still get the maximum on image quality. Depending on the image content (heavily structured or more flat areas) resources for a fixed output quality vary and vice versa (Figure 8). Processing resources have no fixed relationship to output quality and cannot be used for quality estimation. Therefore, output quality measurement becomes an important subject for resource allocation and overall system and application optimization.

Figure 9 shows an SVA with internal quality measurement to indicate its data and budget (resource) dependent performance. Typically, media processing functions can be used to estimate the output quality. In case of sharpness enhancement, the final priority level processed gives an indication of the achieved output quality. For motion estimation, the final, average accuracy and reliability can indicate the output quality.

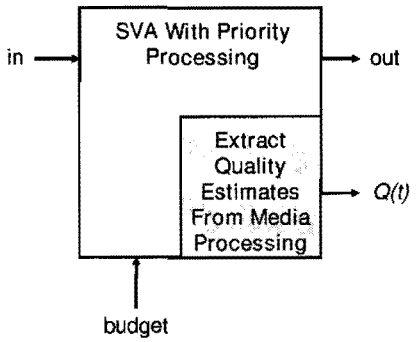


Figure 9: Quality measure within a scalable video algorithm.

3.4 SVA with content dependent resource fluctuations and load balancing

The general schematic diagram of a progress-based resource regulator is shown in Figure 10 [5]. The basic algorithm for media processing contains two new functions for the measurement of progress *P* and used resources *Rr*. These measurements are derived from internal media specific processing data and are therefore independent from external system data. For example, in the 3D-RS motion estimator, *P* is the number of block lines that have been processed and *Rr* is the number of vector candidates that have been used (which is a good measure for the actual number of CPU cycles used [4]).

The external input indicates the available budget per assigned period, typically a frame. For the motion estimator example, this is the total number of vector candidates that is, on average, needed for processing all block lines within a frame.

Together with the measured progress *P*, the expected resource usage can be calculated at incremental periods of a frame. This expected resource usage can then be compared with the measured resource usage *Rr* to calculate the deviation from target *Rd*. The calculated target may be smoothed by a low pass

filter and additionally corrected by a non-linear function before adjusting the resource/quality setting of the scalable media processing algorithm.

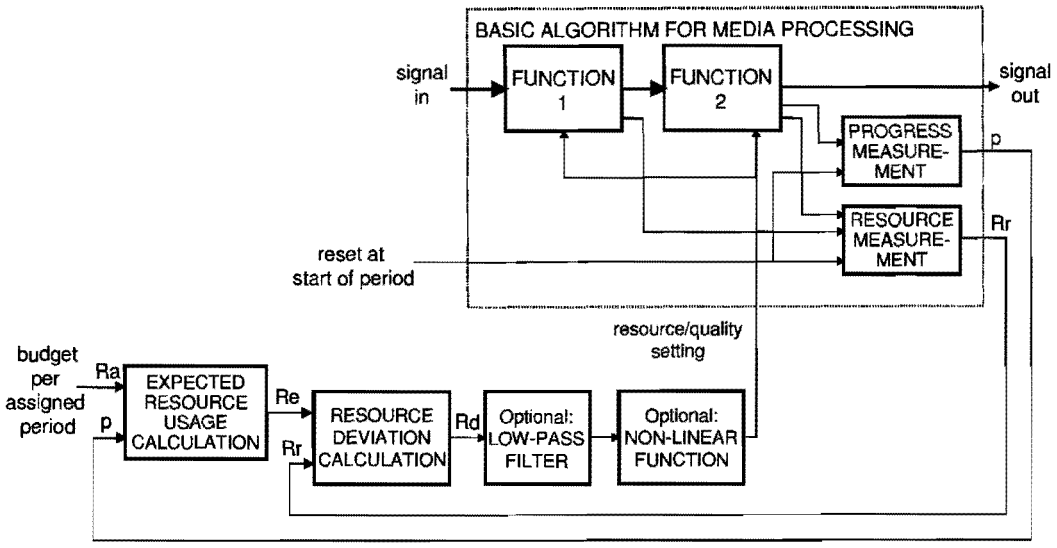


Figure 10: Schematic diagram of the progress-based resource regulator.

This regulation scheme ensures that the regulation properties are independent of the amount of data already processed (actual progress, e.g. screen position). The regulation works on differences between expected and real resource usage during the assigned period and regulates close to a specific resource budget per frame, independent of the input data properties. Despite the very good regulation properties, also a quality gain has been recognized by observers.

4. QUALITY ISSUES IN AN SVA CHAIN

A modular system approach with simple QoS resource management is essential for flexible and manageable multimedia consumer terminals. Consumer terminals include mobile and stationary devices with stand-alone capabilities or within a network environment. Instead of measuring the quality at several positions of a processing chain, SVAs with integrated quality measurement could significantly simplify the system design.

In addition, properties of the input signal can help to optimize the functional structure as well as the quality of the overall system. Input signals come from different sources such as analog NTSC or PAL sources, Y/C, digital sources with different compression algorithms and compression quality (MPEG, DV, H.264...) and so on. These input signals may have specific artifacts such as noise (SNR), blocking, ringing, etc., could vary in resolution (HD, SD, CIF,...) and sampling formats (4:4:4, 4:2:2, 4:2:0, 4:1:1...). Since these input signal properties are important for the entire chain processing, these parameters should be collected in the block 'video analysis' close to the video decoder as depicted in figure 11. Analysis information can then be used for optimizing quality in the resource-scalable processing chain.

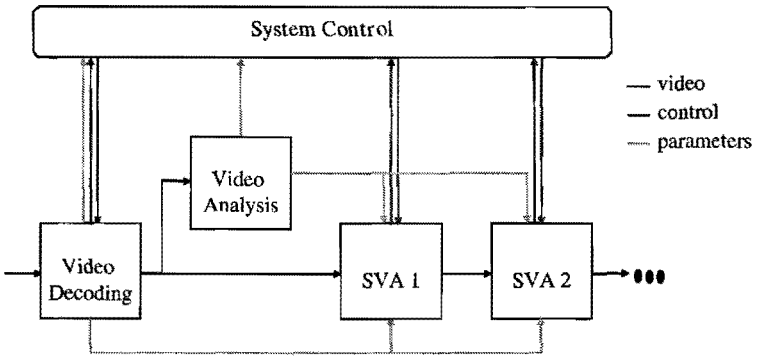


Figure 11: Input signal quality analysis for quality control of the processing chain.

5. CONCLUSIONS

In the past image quality has often been optimized on single processing algorithms which later were combined for the entire application. The result of the final quality often depended on the experience and knowledge of the engineers. The signal source, encoding, and transmission already affects image quality, which should be preserved as much as possible. In the receiver, incoming image quality as well as processing steps including the order of certain algorithms have to be taken into account for optimal image quality. A new approach is the development of scalable video algorithms suitable for programmable components. Simple mapping of algorithms designed for ASIC implementation cannot provide cost-effective and efficient usage of programmable components. Starting from basic SVAs, examples of SVAs for data depending processing resources including a novel approach with load balancing were described. Investigations showed the effectiveness of these methods. Since image quality in scalable applications will dynamically fluctuate, new, real-time methods are required to optimize the overall output quality. Video analysis at an early stage in the processing chain can provide information about the signal source parameters which can be used for optimizing resource distribution over the processing components.

REFERENCES

[1] C. Hentschel, R.J. Bril, M. Gabrani, L. Steffens, K. van Zon, S. van Loo, *Scalable Video Algorithms and Dynamic Resource Management for Consumer Terminals,* Int. Conf. on Media Futures (ICMF 2001), Proceedings, Florence (Italy), May 2001, pp. 193-196.

[2] K. Nahrstedt, H. Chu, S. Narayan, *QoS-aware Resource Management for Distributed Multimedia Applications*, Journal on High-Speed Networking, Special Issue on Multimedia Networking, IOS Press, Vol. 8, No. 3-4, pp. 227-255, 1998.

[3] C. Hentschel, R.J. Bril, Y. Chen, R. Braspenning, Tse-Hua Lan, *Video Quality-of-Service for Consumer Terminals - A Novel System for Programmable Components.* IEEE Transactions on Consumer Electronics Vol. CE-49 (2003), No. 4 (November), pp. 1367-1377.

[4] R. Braspenning, G. de Haan, C. Hentschel, *Complexity Scalable Motion Estimation*, International Conference on Visual Communications and Image Processing (VCIP), Proceedings, San Jose (USA), January 2002, pp. 442-453.

[5] C. Hentschel, R. Wubben, *Novel Resource Regulator for Media-Processing Algorithms on Programmable Components*, International Conference on Consumer Electronics, Digest of Technical Papers, Las Vegas (USA), Jan. 10.-12., 2005, pp. 443-444.

Intelligent Control for Scalable Video Processing

Clemens C. Wüst and Wim F.J. Verhaegh

Philips Research

Increasingly, video processing in digital TVs and set-top boxes is performed in software on programmable components, such as the Philips TriMedia processor. Generally, video processing tasks show strong load fluctuations, which are due to the varying size and complexity of the video data they process. There is often a large gap between the worst-case and the average-case resource needs of a video processing task. We present an approach that allows close-to-average-case resource allocation to a single video processing task, based on asynchronous, scalable processing, and QoS adaptation. A scalable video processing task can reduce its processing needs by decreasing the quality level of processing, at the level of individual video frames. The QoS adaptation balances different QoS parameters that can be tuned, based on user-perception experiments: the quality level at which frames are processed, deadline misses, and changes in the quality level between successive frames. We model the balancing problem as a stochastic decision problem, and propose two intelligent control strategies, based on a Markov decision process and reinforcement learning, respectively. We validate our approach by means of simulation experiments, and conclude that both strategies perform close to optimum.

PHILIPS

Intelligent Control for Scalable Video Processing

Clemens Wüst, Wim Verhaegh

Philips Research Laboratories, the Netherlands

PHILIPS

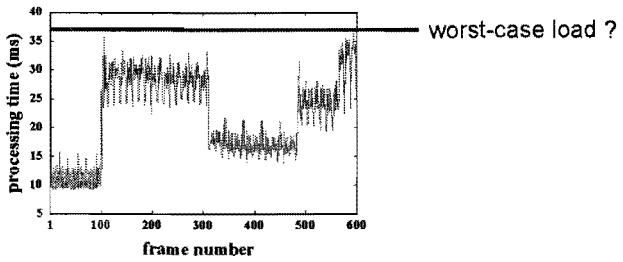
Overview

PHILIPS

Introduction

- Video processing in software
  - User expects high-quality output
  - Deadlines on the completion times of frames

- Many video algorithms show a highly fluctuating load



- Given a fixed processing-time budget, lower than worst-case load

How can we make the best of it?

PHILIPS

Introduction

Our Approach

1. Asynchronous, work-preserving processing
   - Using frame buffers

2. Scalable Video Algorithm (SVA)
   - Frames can be processed at different quality levels
   - Trade-off picture quality and processing needs

3. Soft real-time task, hence we allow occasional deadline misses

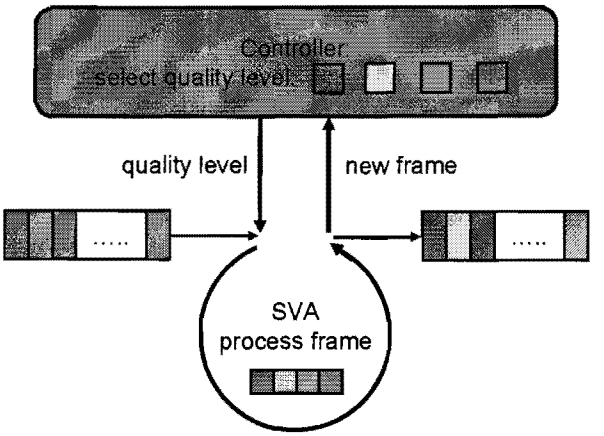
4. QoS trade-off
   - Deadline misses
   - Picture quality
   - Quality fluctuations

} QoS measure reflects user-perceived quality

PHILIPS

QoS Control Problem

SVA-Controller Interaction

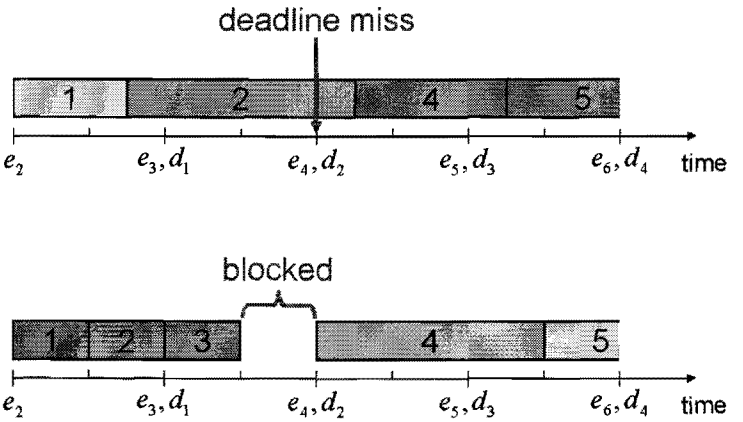


5

PHILIPS

QoS Control Problem

Real-Time Processing Example



6

PHILIPS

QoS Control Problem

Revenue for Processed Frame

Sum of:

- reward for selected quality level
- penalty for each deadline miss
- penalty for changing the quality level

Revenue
8 - 10,000 - 100 = -10,092

Current quality level

q_0	q_1	q_2	q_3
4	6	8	10

Deadline miss: 10,000

Current quality level

Previous quality level	q_0	q_1	q_2	q_3
q_0		10	100	1,000
q_1	50		10	100
q_2	500	50		10
q_3	5,000	500	50	

7

PHILIPS

QoS Control Problem

QoS Measure

- Average revenue per frame
- Reflects the user-perceived quality, provided that the revenue parameters are well chosen

QoS Control Problem

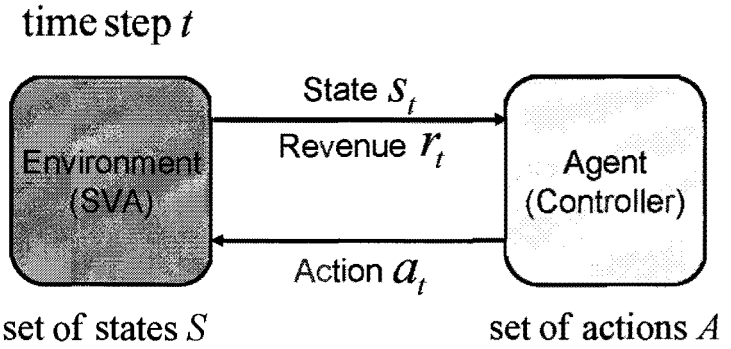
- At each decision point, select the quality level
- Goal: maximize the QoS measure
- Difficult on-line problem: what will the future bring?

8

PHILIPS

Reinforcement Learning

Agent-Environment Interaction



9

PHILIPS

Reinforcement Learning

Agent's Goal

- Maximize the expected return
- Return at time step *t*

infinite time horizon

$$R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1}$$

discount parameter

10

Markov Decision Process

- We assume a memoryless state signal:
  state s_t and action a_t provide sufficient information to predict state s_{t+1} and revenue r_{t+1}

- Hence, the reinforcement learning task is a Markov Decision Process (MDP)

- We assume a finite MDP:
  - Finite set of states S , finite set of actions A
  - One-step dynamics:

$$p^a_{ss'} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$$

$$r^a_{ss'} = E\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\}$$

Taking Actions

- To select actions, the agent implements a policy

- A stochastic policy π provides for each state $s \in S$ and for each action $a \in A$ a probability $\pi(s,a)$ of selecting action a in state s

- A deterministic policy π provides for each state $s \in S$ a single action $\pi(s) \in A$ to be chosen

- Without loss of optimality, we can restrict ourselves to deterministic policies

PHILIPS

Reinforcement Learning

Value Functions

- State value of state $s \in S$ under policy π

$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\}$$

- Action value of state $s \in S$ under policy π

$$Q^{\pi}(s,a) = E_{\pi}\{R_t \mid s_t = s, a_t = a\}$$

- A policy π is better than or equal to a policy π' if

$$V^{\pi}(s) \geq V^{\pi'}(s) \quad \forall s \in S$$

- We are looking for an optimal deterministic policy

13

PHILIPS

Reinforcement Learning

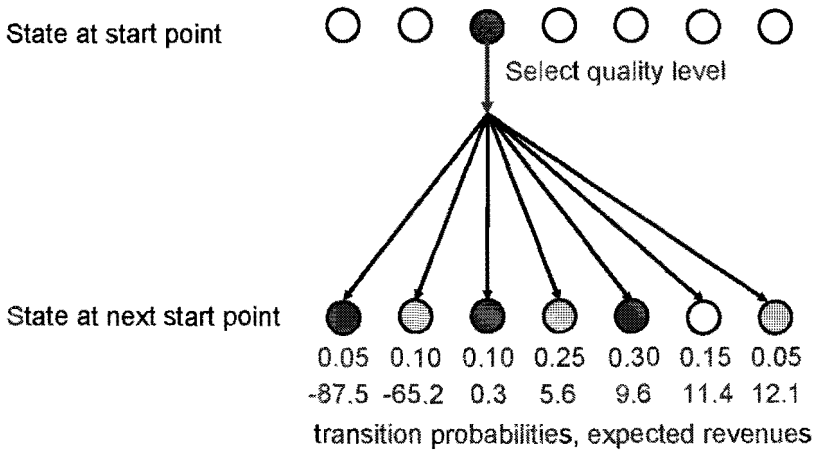
Solution Approach

- Compute an optimal policy OFFLINE (= before $t = 0$)
  - Requires transition probabilities $p^a_{ss'}$
  - Requires expected revenues $r^a_{ss'}$
  - Algorithms: policy iteration, value iteration, ...
- Compute an optimal policy ONLINE, at the discrete time steps, using the experienced states and revenues
  - Algorithms: SARSA, Q-Learning, ...

14

PHILIPS

Offline Approach



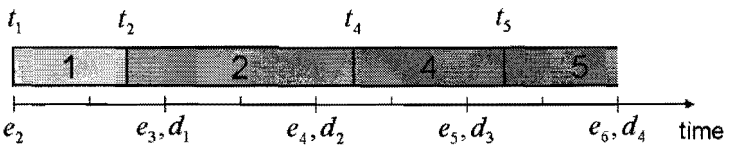
- If the transition probabilities and expected revenues are known, an optimal policy can be computed

15

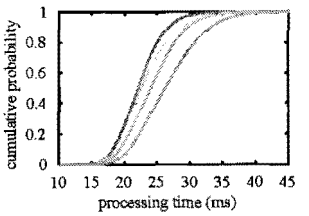
PHILIPS

Offline Approach

- Progress: measure for the amount of budget left until the deadline of the frame to be processed



- State
  - progress interval (discrete!)
  - previous quality level
- Action = select quality level
- Transition probabilities and expected revenues: computed using processing-time statistics

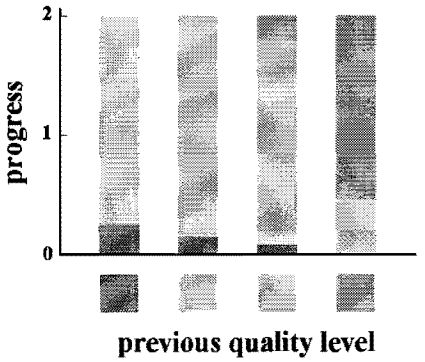


16

PHILIPS

Offline Approach

- Given this model, we use the value iteration algorithm to compute an optimal policy for a particular value of the budget
- This approach results in a low run-time overhead



17

PHILIPS

Online Approach

Q-Learning

- Based on learning action values (Q-values)
- State
  - progress (continuous!)
  - previous quality level
- Action = select quality level
- The agent start with randomly chosen action values

18

PHILIPS

Online Approach

Q-Learning

- At each decision point $t > 0$
  - Given the state transition $s_{t-1} \rightarrow s_t$, action a_{t-1}, and revenue r_t, the controller first updates Q-value $Q(s_{t-1}, a_{t-1})$

$$Q(s_{t-1}, a_{t-1}) = (1-\theta)Q(s_{t-1}, a_{t-1}) + \theta(r_t + \gamma \max_{a \in A} Q(s_t, a))$$

  - Next, given state s_t, the controller selects action a_t for which $Q(s_t, a_t)$ is maximal

- Default: at each decision point one action value is updated;

- Exploring actions are needed

19

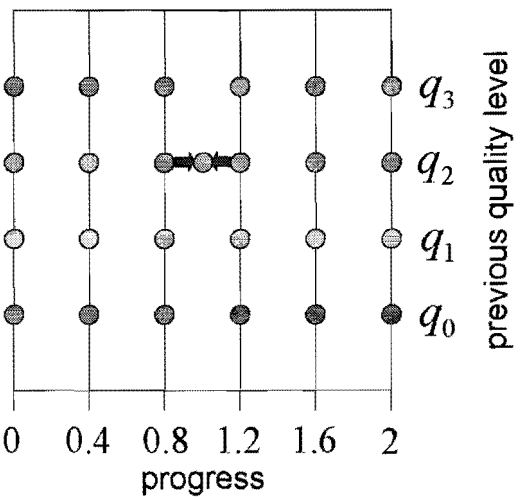
PHILIPS

Online Approach

Continuous state space

- We learn action values only for a finite set of gridpoint states

- To select the quality level, given the state, we interpolate between the learned action values for gridpoint states



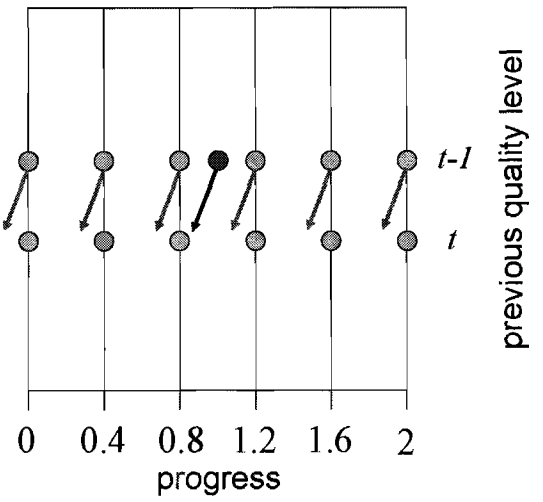
20

PHILIPS

Online Approach

Learning

- Progress increase delta
- We can calculate the effect of delta for all gridpoint states
- Hence, we can update action value $Q(s, a_{t-1})$ for all gridpoint states s



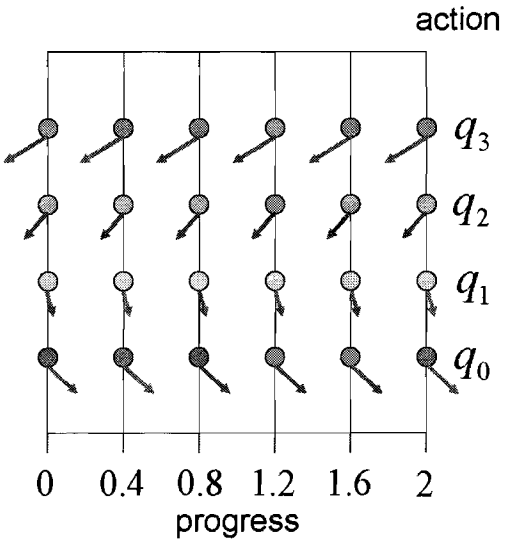
21

PHILIPS

Online Approach

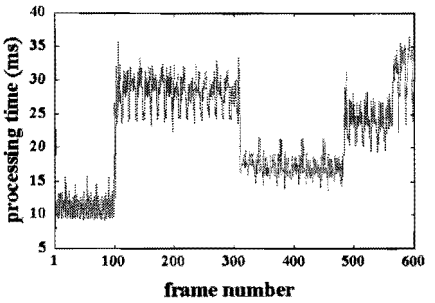
Learning (ctnd)

- Estimate the effect of other actions
- Hence: all action values updated in each step!
- We no longer need exploring actions



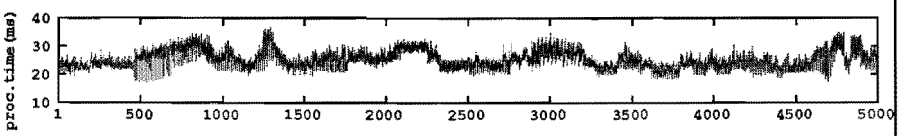
22

- Both approaches implicitly assume that processing times of successive frames are mutually independent
- TRUE for stochastic load fluctuations
- NOT TRUE for structural load fluctuations

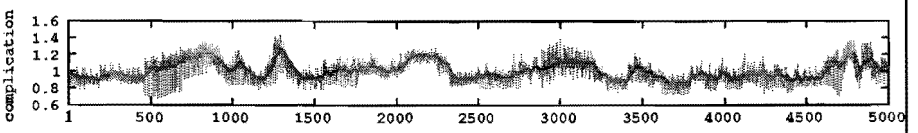


- As a result, both approaches perform sub-optimal

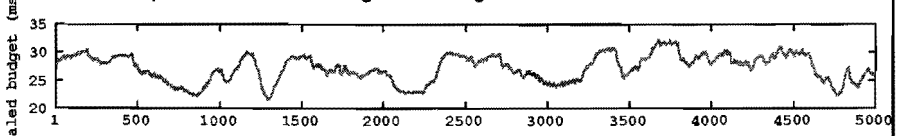
Scaled Budget Enhancement



1. At each decision point, compute the complication factor
   = *proc.time of frame / expected proc.time for applied quality level*



2. Filter out the stochastic load fluctuations
3. Compute the scaled budget = *budget / structural load*



PHILIPS

Handling Load Fluctuations

Scaled Budget Enhancement

Adapt offline strategy

- Compute a policy for many different values of the scaled budget
- During run time, at each decision point:
  - Compute the scaled budget
  - Compute the state of the SVA
  - Apply the policy corresponding to the scaled budget, and use the state to select the quality level
  - Interpolate between policies

Adapt online strategy

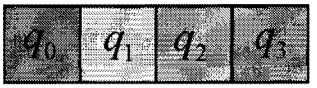
- Add scaled budget directly to the state

25

PHILIPS

Simulation Experiments

- Scalable MPEG-2 decoder
  - TriMedia 1300-180MHz platform
  - Quality levels (based on IDCT pruning):



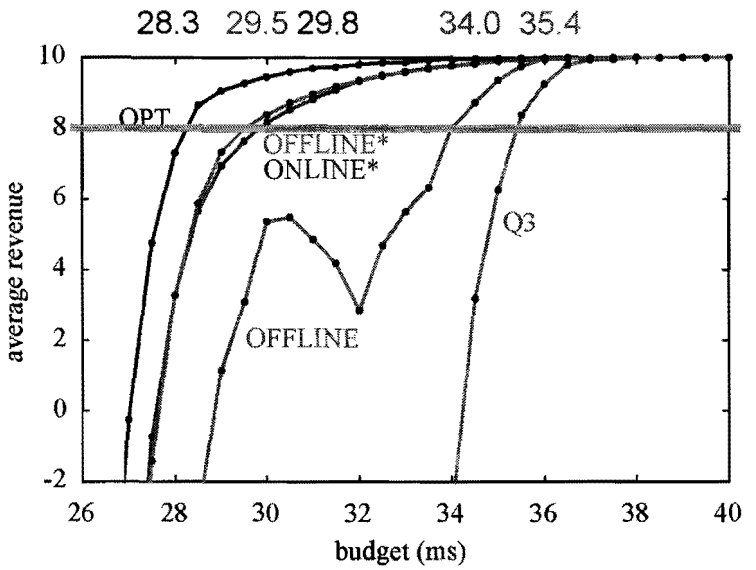
  - Sequence 'TV' (five episodes of 'Allo 'Allo, 230,936 frames, 2.5 hours)
- Latency: 3 periods (= work ahead of at most 2 periods)
- Revenues: based on input of video experts (slide 8)
- Control strategies: OFFLINE, OFFLINE*, ONLINE*, Q0,...,Q3
- For each control strategy, we simulate processing sequence 'TV' for a fixed value of the processing-time budget

26

PHILIPS

Simulation Experiments

Average Revenue

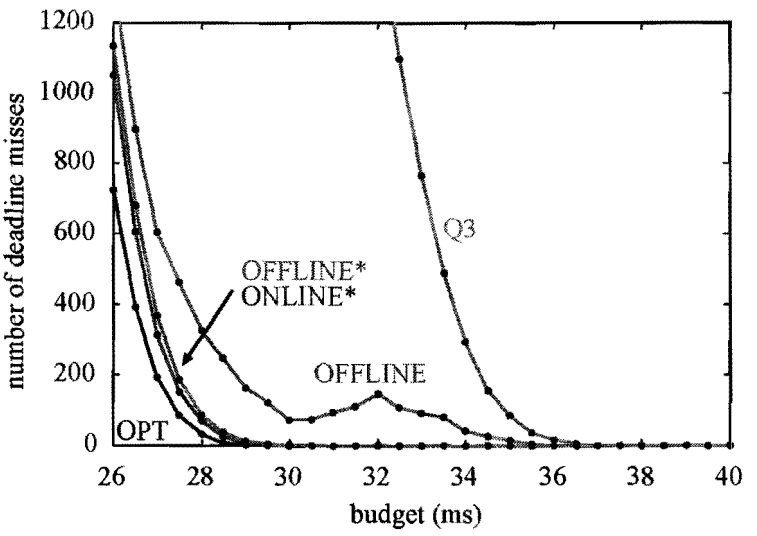


27

PHILIPS

Simulation Experiments

Deadline Misses

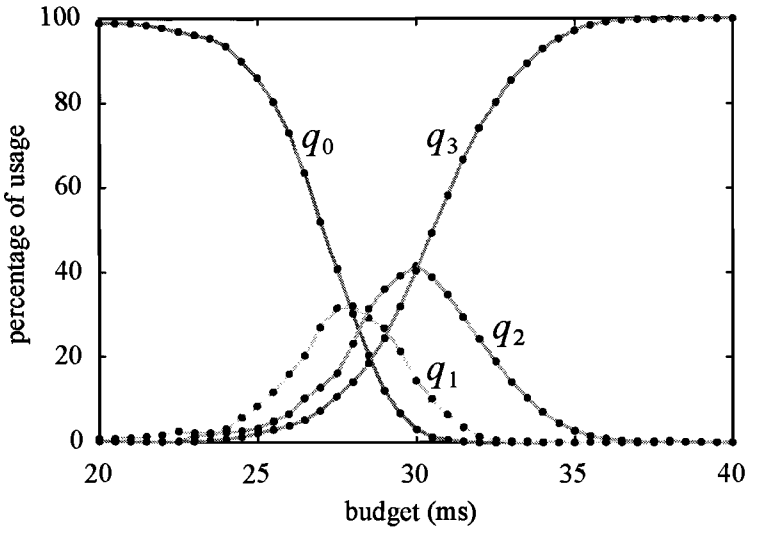


28

PHILIPS

Simulation Experiments

Quality-Level Usage

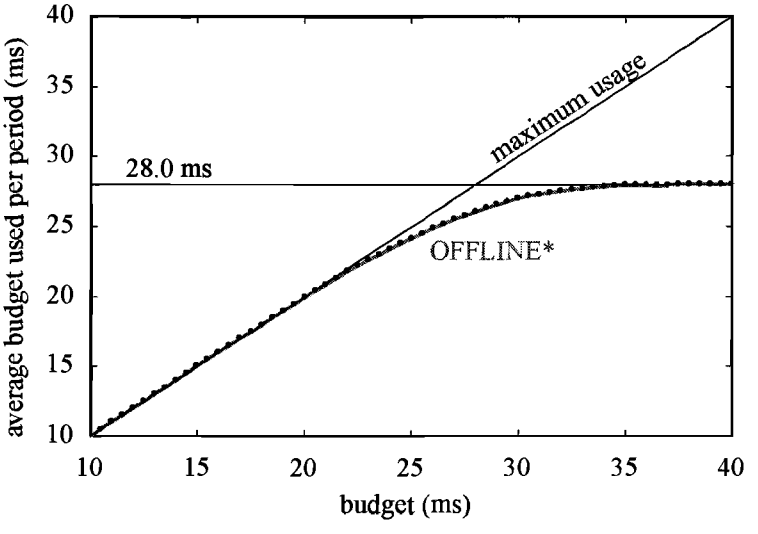


29

PHILIPS

Simulation Experiments

Budget Usage



30

Cross-Trace Simulations

sequence:	tv	mov	soc
sensible budget range:	27.5–33 ms	25–30.5 ms	25–28.5 ms
OPT	8.92	8.65	8.69
Q0	3.22	3.53	2.89
Q1	1.39	−2.35	−1.85
Q2	−25.09	−46.92	−24.27
Q3	−207.43	−483.09	−151.61
OFFLINE(tv)	1.19	7.27	5.75
OFFLINE(mov)	−50.69	4.81	2.38
OFFLINE(soc)	−80.69	0.68	−1.33
OFFLINE*(tv)	7.24	7.39	7.00
OFFLINE*(mov)	6.46	7.53	5.94
OFFLINE*(soc)	6.99	7.06	6.83
ONLINE*(tv)	7.15	7.54	6.51
ONLINE*(mov)	7.09	7.46	6.46
ONLINE*(soc)	7.17	7.41	6.59

Conclusion

- Problem
  - Video processing algorithm with highly fluctuating load
  - Fixed processing-time budget, lower than worst-case needs
  - How to optimize the user-perceived quality?
- Approach
  - Asynchronous work-preserving processing
  - Scalable video algorithm
  - QoS trade off: deadline misses, processing quality, quality fluctuations
- Control strategies
  - Offline, online, scaled budget enhancement
- Simulation experiments
  - OFFLINE* and ONLINE* perform close to optimum
  - OFFLINE* and ONLINE* are independent of the applied statistics

PHILIPS

Conclusion

Acknowledgements

- Reinder Bril
- Christian Hentschel
- Laurentiu Papalau
- Clara Otero Perez
- Liesbeth Steffens
- Rob Wubben

Video streaming over wireless network

Peter D.V. van der Stok

Philips Research, TU/e

Wireless media like IEEE 802.11a and 802.11b and IEE 802.11g are sensitive to perturbations. Packets are easily lost and the bandwidth of the medium changes rapidly. The effect on the video streamed over a wireless medium is disastrous. In this talk the effects on the video quality are shown to depend on the operational conditions: the transmission protocol and the video source. It is shown how the deployment of SNR scalable and temporal scalable video reduces the effects of the transmission perturbations. Key is a controlled adaptation of the stream at the sender. The result is that the highest possible video is transmitted over the wireless line and that under packet loss the user never perceives any artifacts, but only a reduced quality video with possibly a visible gap between two successive frames.

PHILIPS

PHILIPS

Video Streaming over Home Network

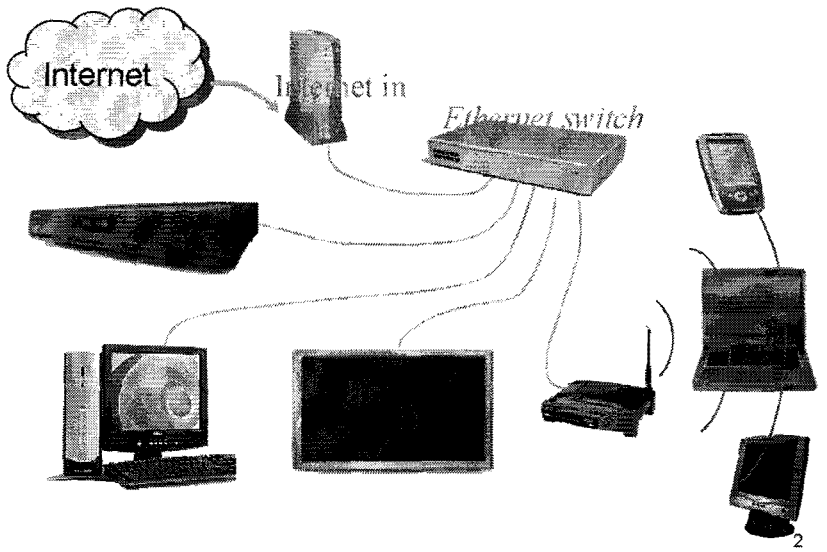
Peter van der stok

31 March 2005

Technical University Eindhoven

PHILIPS

Home network example



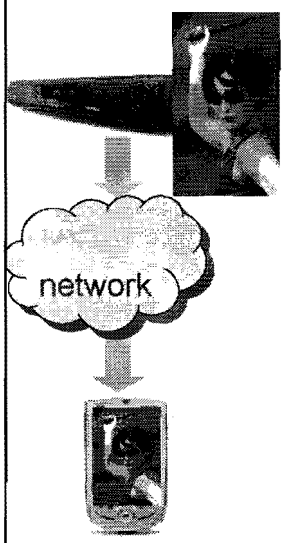
PHILIPS

Contents

1. Video over home network
2. Transport driven video artifacts
3. Techniques to remove artifacts

PHILIPS

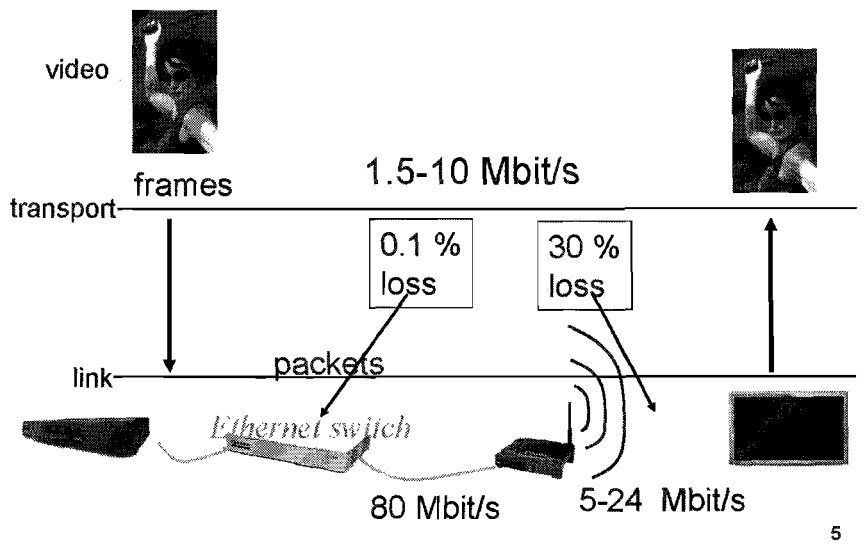
QoS chains



Quality of video	Size of video bit/s
Quality of network	Bandwidth, delay
Quality of renderer	Processing power
Quality of experience	Perception by user

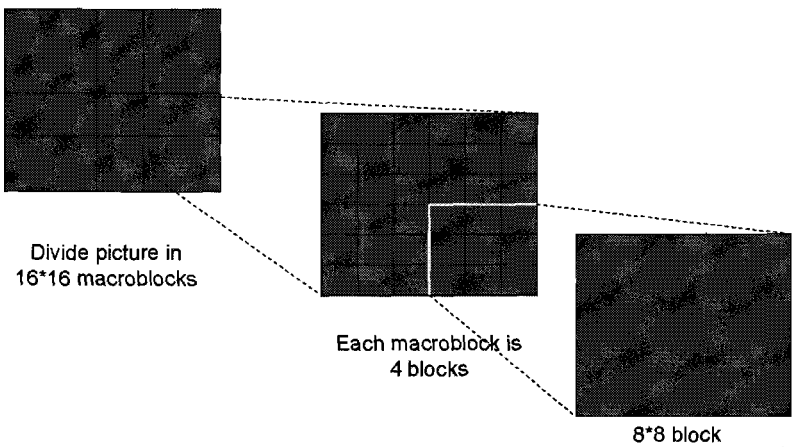
PHILIPS

Network stream



PHILIPS

MPEG2 video



PHILIPS

MPEG2 video

Frame types

- I frames

- P frames

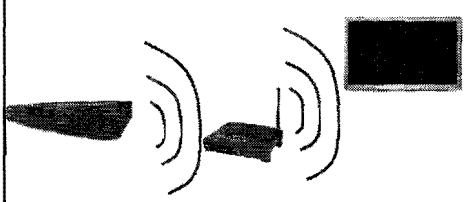
- B frames

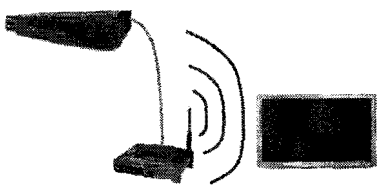
- Group of Pictures (GOP): Set of frames between two I frames
  - IBBPBBPBBI or IPPPI, or II

PHILIPS

Wireless video streaming

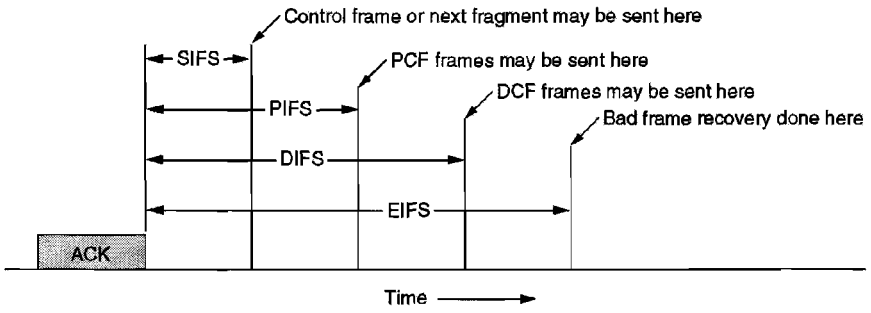
- Video frame contains 20-60 packets
- With 30% loss no frame will ever be displayed
- Standard foresees retransmission from point to point





PHILIPS

IEEE 802.11 Protocol



Interframe spacing in 802.11.

Courtesy of A. Tanenbaum

PHILIPS

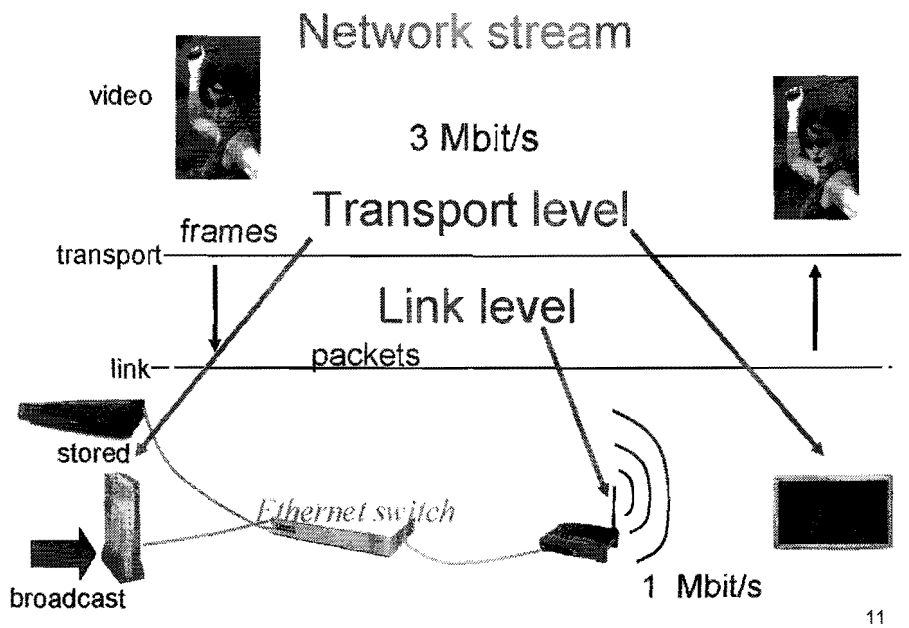
IEEE 802.11 Link QoS

- Every received packet is immediately acknowledged
- Unacknowledged packet is repeated
- sending failed after n unacknowledged repetitions

$0 < n < 256$

Conclusion: under packet-loss, bandwidth decreases

PHILIPS



11

Research, Peter van der Stok, 31 March 2005

PHILIPS

Contents

1. Video over home network
2. Transport driven video artifacts
3. Techniques to remove artifacts

12

Research, Peter van der Stok, 31 March 2005

Two Internet Protocols

Transmission Control Protocol, and Real-Time Transport Protocol

TCP

A receiver that accepted a packet, accepted all preceding packets in the sending order

RTP

All accepted packets are consumed in time

The Real-Time Transport Protocol

UDP header

Source port	Destination port
UDP length	UDP checksum

32 bits

Control info	Sequence number
TimeStamp	
Synchronization source identifier	

32 bits

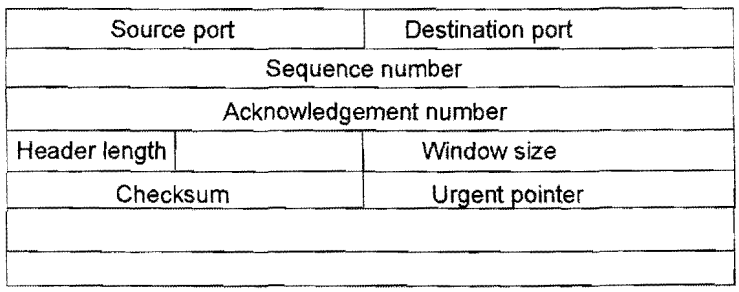
Courtesy of A. Tanenbaum

PHILIPS

The Transmission Control Protocol

TCP header

32 bits



Courtesy of A. Tanenbaum

PHILIPS

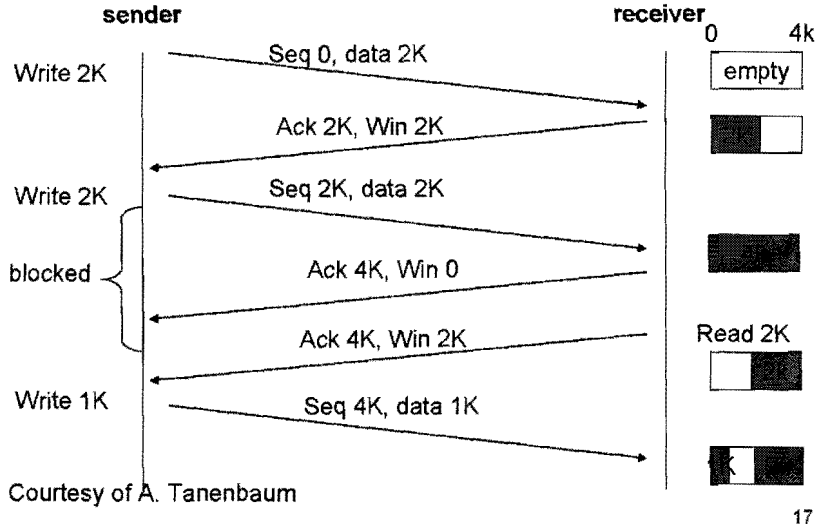
TCP (2)

Separation of acknowledgements and permission to send
leads to variable sized windows

Use of selective repeat (NAK)

PHILIPS

TCP (3)



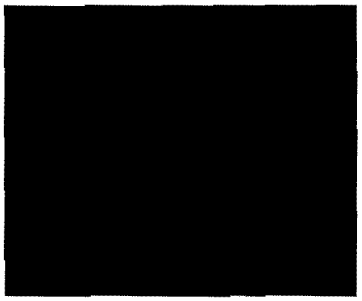
PHILIPS

Single loss

TCP

smooth

RTP

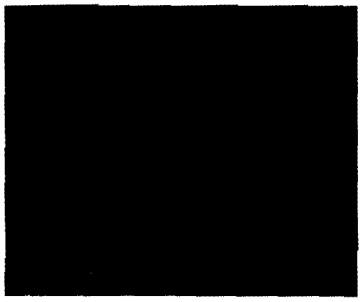


Intermittent losses

PHILIPS

Bursty loss

TCP



Intermittent delays

RTP

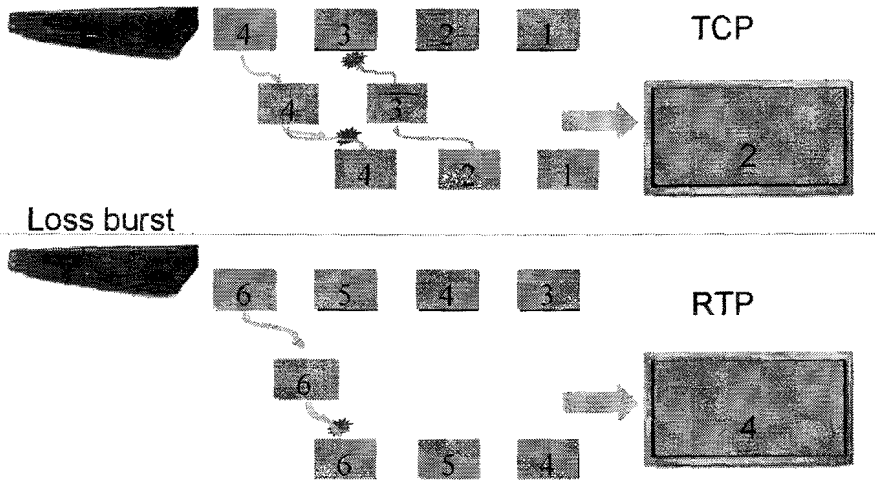
Intermittent losses

19

Research, Peter van der Stok, 31 March 2005

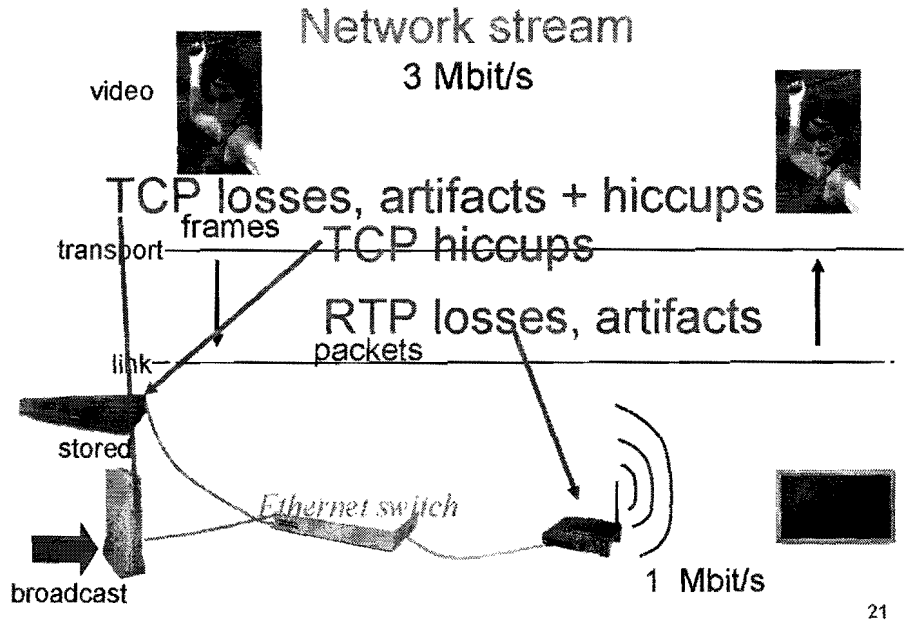
PHILIPS

TCP hiccup



20

Research, Peter van der Stok, 31 March 2005

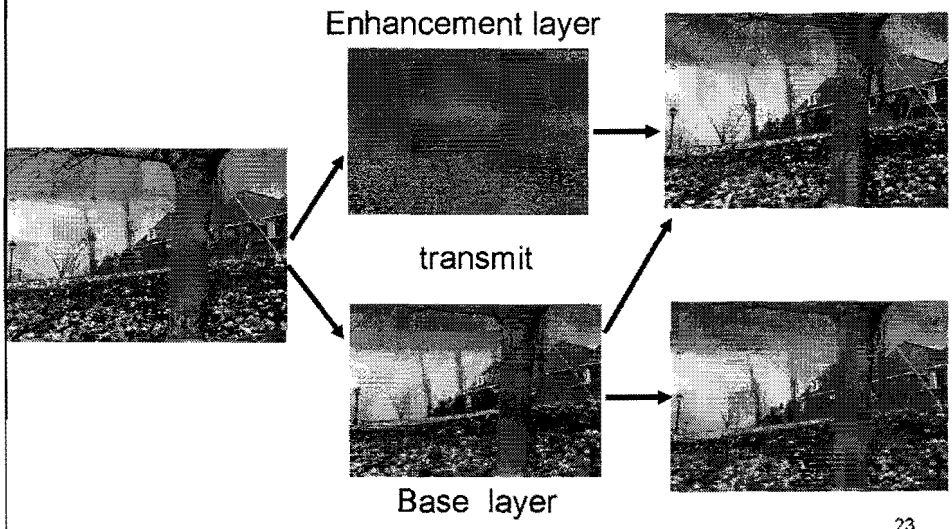


Contents

1. Video over home network
2. Transport driven video artifacts
3. Techniques to remove artifacts

PHILIPS

Scalable video code



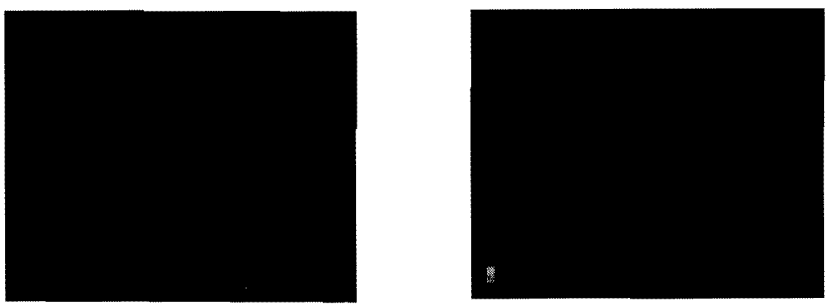
Research, Peter van der Stok, 31 March 2005

23

PHILIPS

Perturbed wireless streaming

Microwave on



RTP unlayered

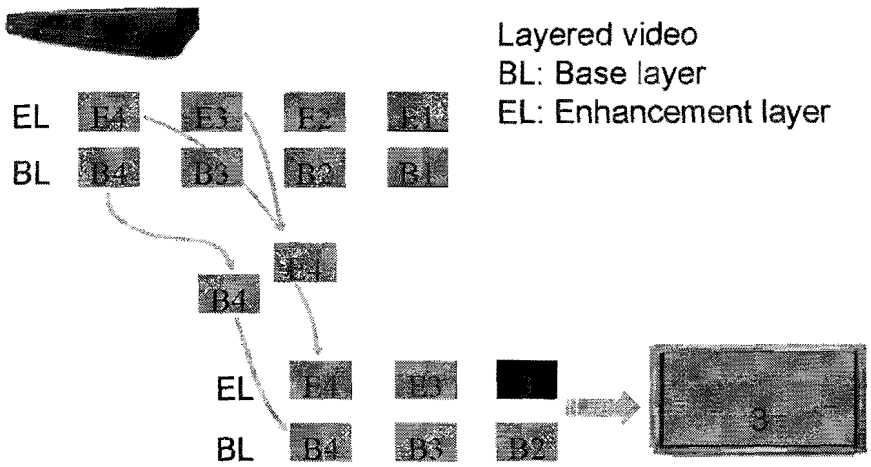
RTP layered

Research, Peter van der Stok, 31 March 2005

24

PHILIPS

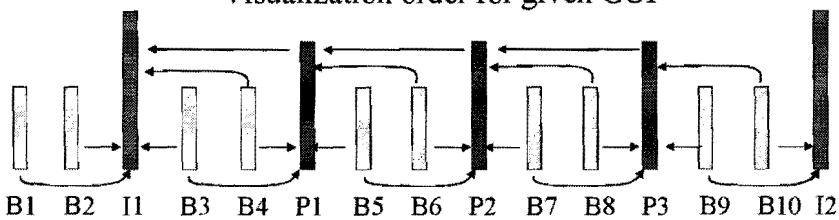
Sending layers



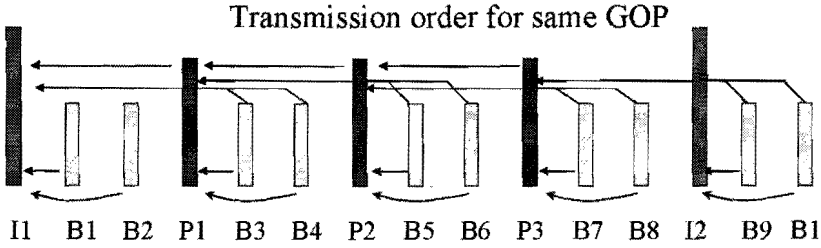
PHILIPS

Group of Pictures (GOP)

Visualization order for given GOP

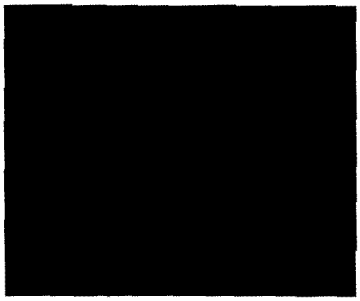


Transmission order for same GOP

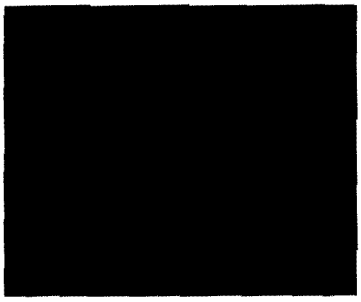


PHILIPS

I-Frame Delay (IFD)



I-frames lost



I-frames delayed

PHILIPS

IFD protocol
Sufficient bandwidth

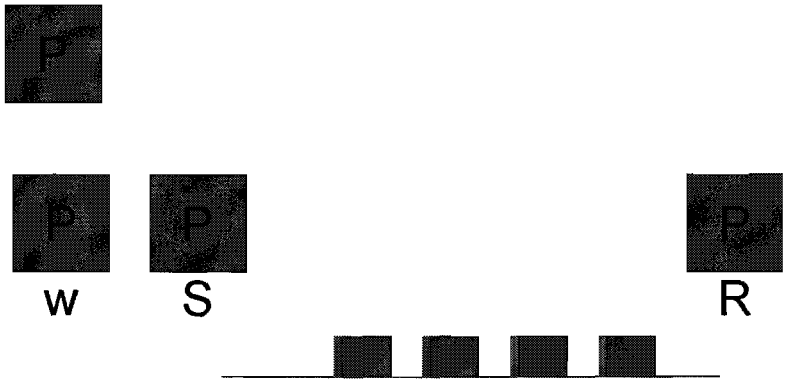
W

S

R

PHILIPS

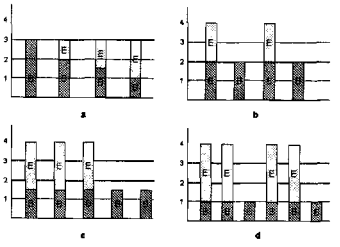
IFD protocol
Insufficient bandwidth



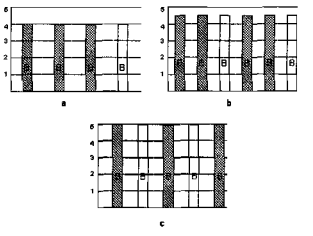
PHILIPS

User tests

- 20 USI students
- rate videos with identical bit-rate
  but different bit distributions



1



2

User test conclusions

Skipped frames in enhancement layer has small impact (1)
Skipped frames in base layer has significant impact (2)
Best SNR results with largest base layer

Steady state measurements
Repeat for fluctuating conditions

Quality-of-Experience in wireless multi-standard multi-media devices: a design-time/run-time cross-layer approach

Wolfgang Eberle

IMEC Leuven

Wireless communications has seen a tremendous diversification in applications and growth in the number of users in the last decade. Two types of terminals have evolved: software-defined (SDR) or software-reconfigurable radios (SRR), capable of handling multi-standard multi-mode multi-service applications, and very dedicated ultra-low power radios for e.g. sensor networks or RFID tags. Focusing on the SDR/SRR path, flexibility requires these radios to adapt optimally to changing quality-of-service (QoS) demands of the user and to a dynamic environment respecting a limited amount of available energy resources. The traditional early divide & conquer design approach that led to independent design of RF front-end, digital baseband, and protocol layers has proven to result in rather high design margins and hence low energy efficiency on the average and little adaptivity to service or channel dynamics. In recent years, cross-layer and mixed-signal design concepts have been flourishing. We illustrate recent design concepts and their succesful application in the context of wireless LAN (both for single- and multiple antennas, single and multiple users) with - currently - MPEG-4 video streaming as driver application.

Quality-of-Experience in wireless multi-standard multi-media: a design-time/run-time approach

W. Eberle*, B. Bougard*, S. Pollin*, C. Blanch, G. Lenoir

IMEC Wireless Research
Contact: wolfgang.eberle@imec.be

***also KU Leuven**

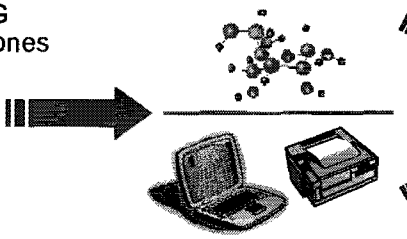
SEEDS FOR TOMORROW'S WORLD

© imec 2006

Towards ubiquitous communications and true multimedia services

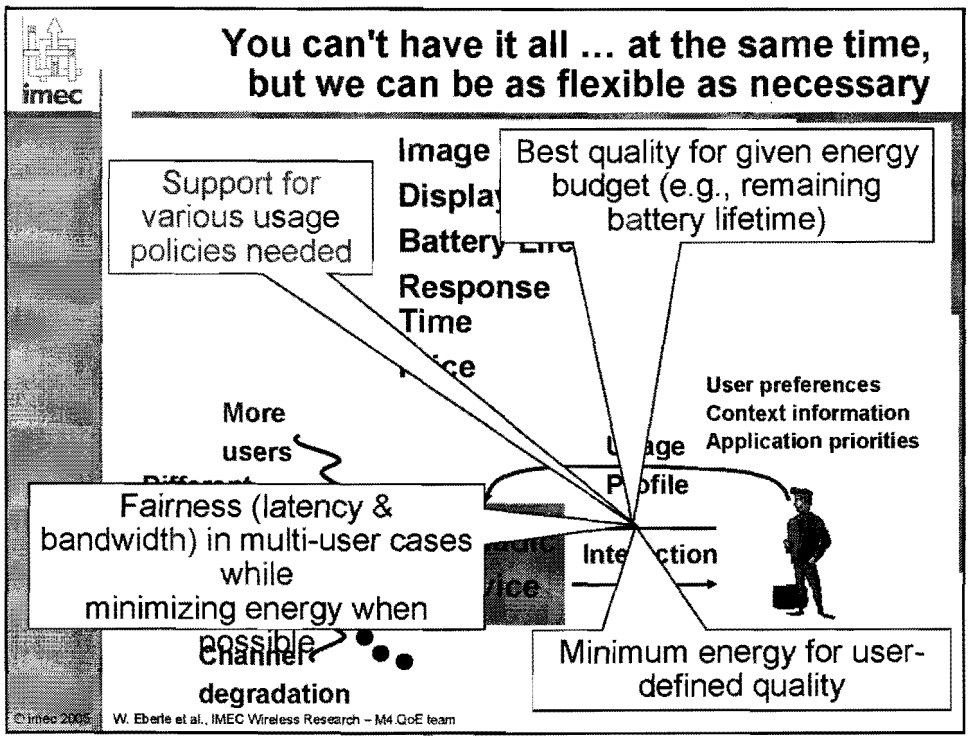
Ubiquitous availability of services requires *ubiquitous, wireless communication*

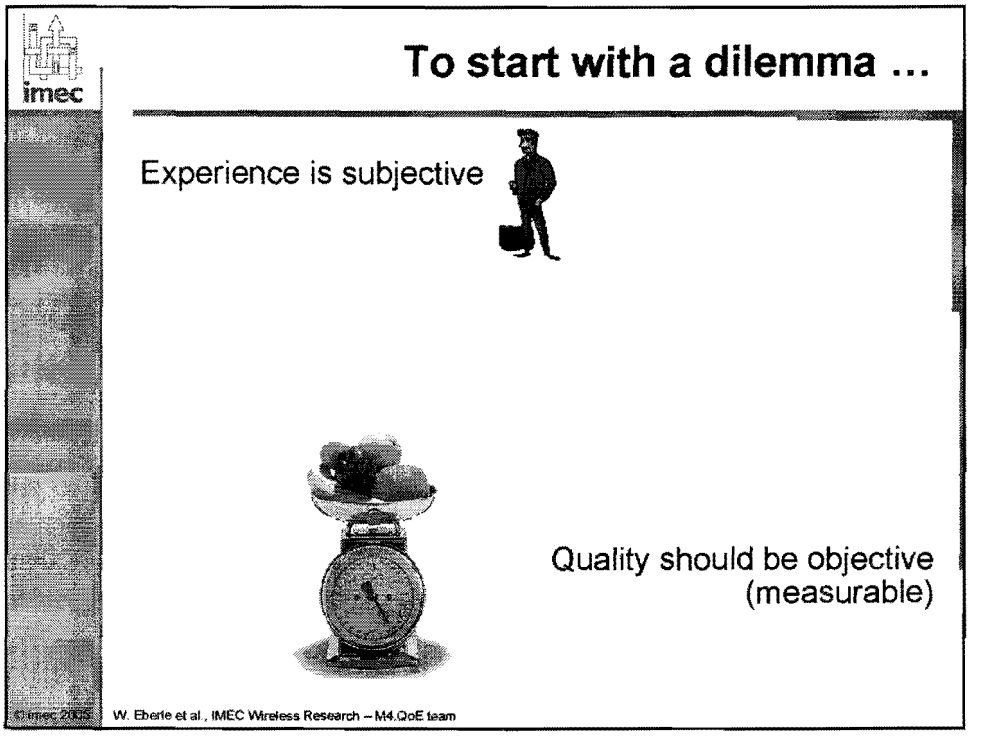
2G/2.5G Mobile Phones



M4 terminals

© imec 2006


imec
You can't have it all ... at the same time, but we can be as flexible as necessary
Support for various usage policies needed
Image
Displa
Battery
Response Time
Best quality for given energy budget (e.g., remaining battery lifetime)
User preferences
Context information
Application priorities
More users
Fairness (latency & bandwidth) in multi-user cases while minimizing energy when possible
Channel degradation
Minimum energy for user-defined quality
© imec 2005
W. Eberle et al., IMEC Wireless Research – M4.QoE team


imec
To start with a dilemma ...
Experience is subjective
Quality should be objective (measurable)
© imec 2005
W. Eberle et al., IMEC Wireless Research – M4.QoE team

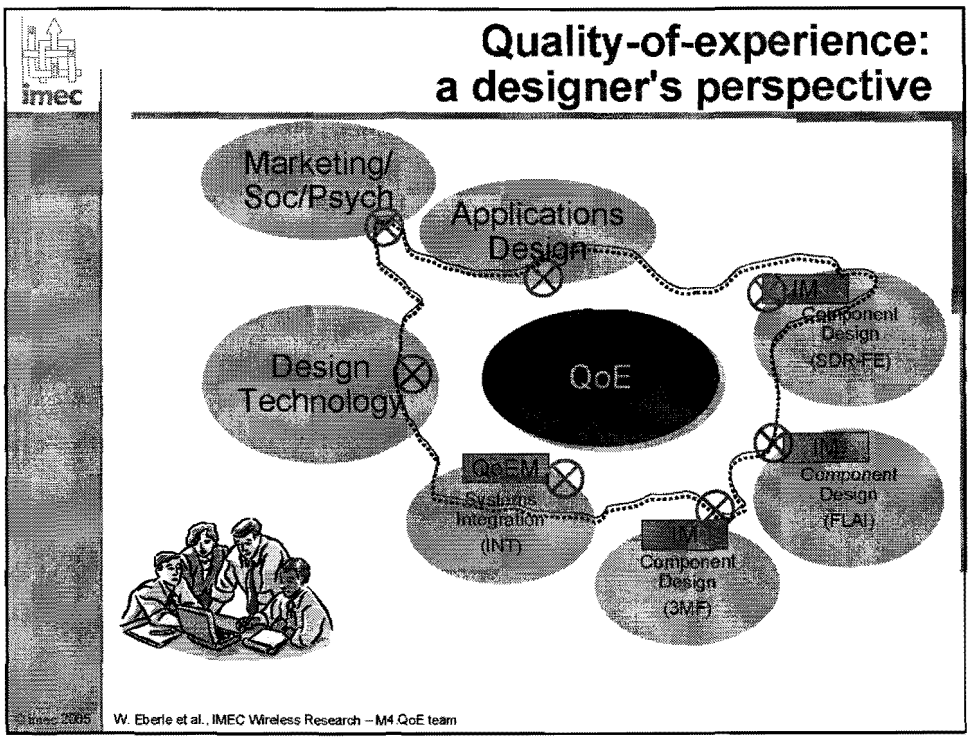
Service or experience: QoS vs. QoE

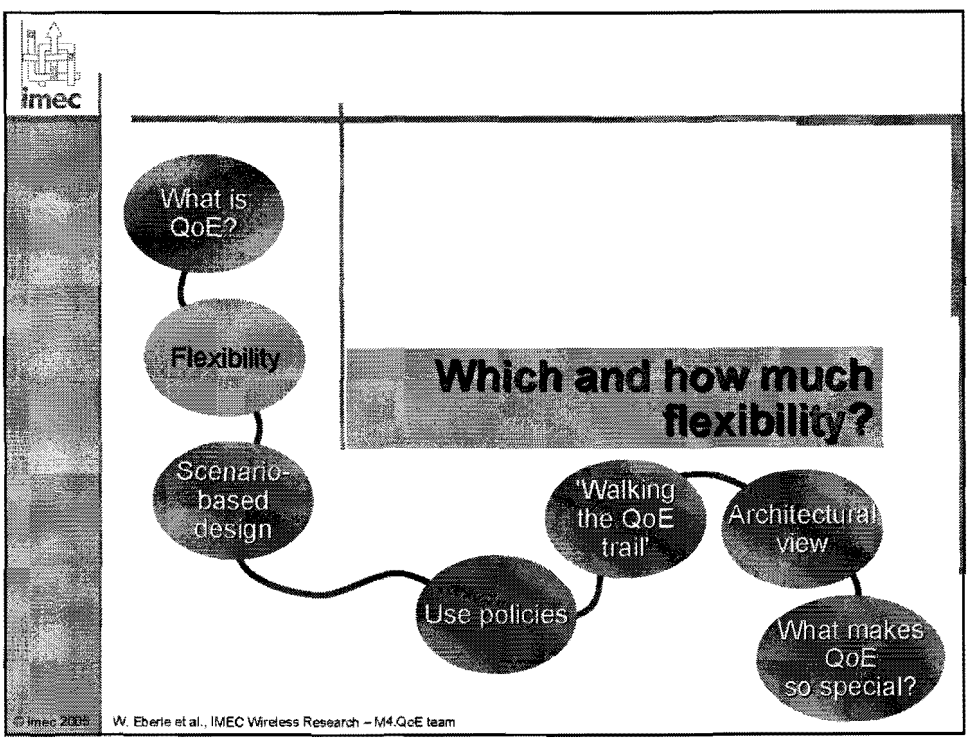
IST NEWCOM network-of-excellence defines QoE related to QoS:

Quality-of-experience (QoE) is an important extension to QoS on a per-service basis: QoE acts as an umbrella covering differing QoS requirements for various services and environment contexts allowing the user to specify more generic policies and priorities ('go for maximum visual quality; go for longest battery life time; etc.)

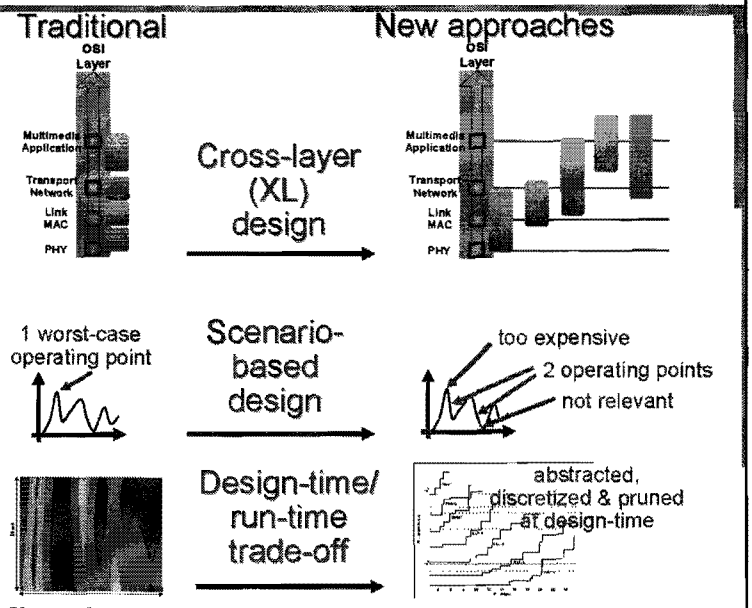
Quality-of-experience: a user perspective

- e.g. R. Jain in IEEE Multimedia, 2004 on Quality of experience:
  - QoS-based system design is popular in related areas
  - Can we extend QoS to QoE?
  - Experience is subjective
  - Experience is context-dependent
  - QoS considers factors that are objective
  - We must develop measures that will help us capture QoE in a given application and use it
  - These systems need new approaches to define performance measures considering the subjective nature of the system's most important component – the user

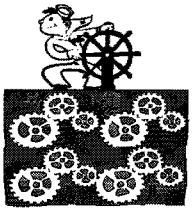

Quality-of-experience:
a designer's perspective
Marketing/
Soc/Psych
Applications
Design
Design
Technology
QoE
M4
Component
Design
(SDR-FE)
QoE-SM
System
Integration
(INT)
M1
Component
Design
(3MF)
M3
Component
Design
(FLAI)
© imec 2005
W. Eberle et al., IMEC Wireless Research – M4.QoE team


What is
QoE?
Flexibility
Which and how much
flexibility?
Scenario-
based
design
'Walking
the QoE
trail'
Architectural
view
Use policies
What makes
QoE
so special?
© imec 2005
W. Eberle et al., IMEC Wireless Research – M4.QoE team

Can we improve our design approach?



Key idea: Adaptive resource management



Cross-layer design

- Optimize the system across classical layers, exchange information across layers and reduce overhead and inefficiencies
- Determine the relevant 'knobs' of the system

Scenario-based design
Design for different goals:

- Low average energy and/or low peak power consumption
- Fairness across multiple users
- Best quality for a given power consumption
- Maximum capacity for minimum required performance
- ...

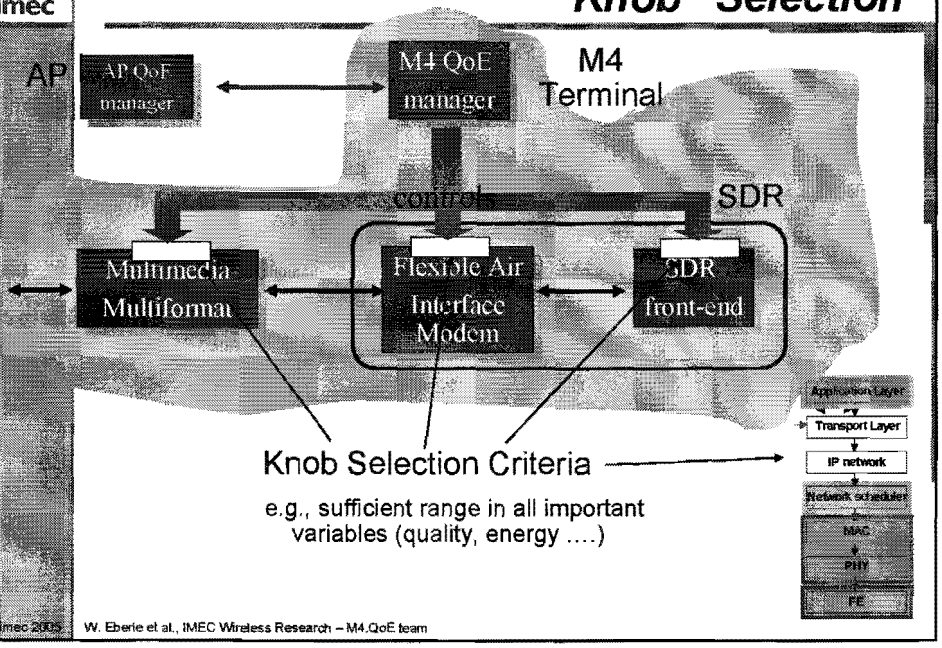
Trade-off design/calibration/run-time

- Optimize at design-time for all static/predictable effects to minimize implementation cost

Component flexibility

- Three ways to achieve flexibility in a component
  - Functional
    - e.g. different algorithms trading off performance vs. complexity
  - Architectural
    - e.g. voltage, frequency, alternative signal paths
  - Mapping
    - e.g. different utilization of the available resources (MPSoC)
- We call such 'flexibility parameters' *knobs*

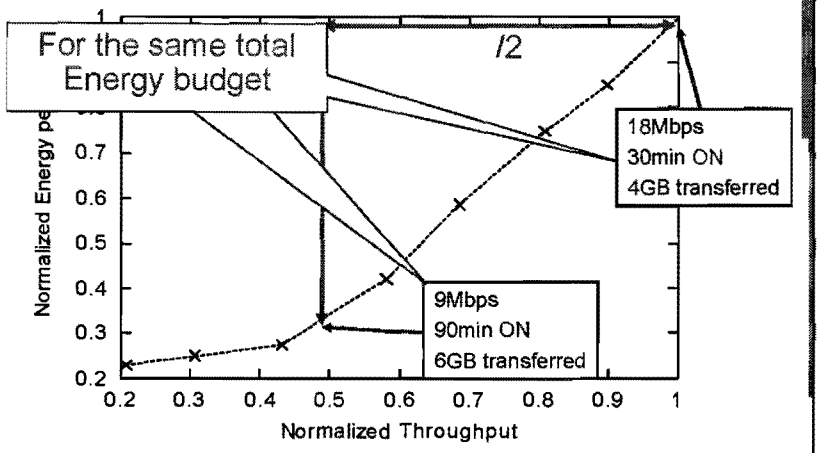
Proposed QoE Architecture *"Knob" Selection*

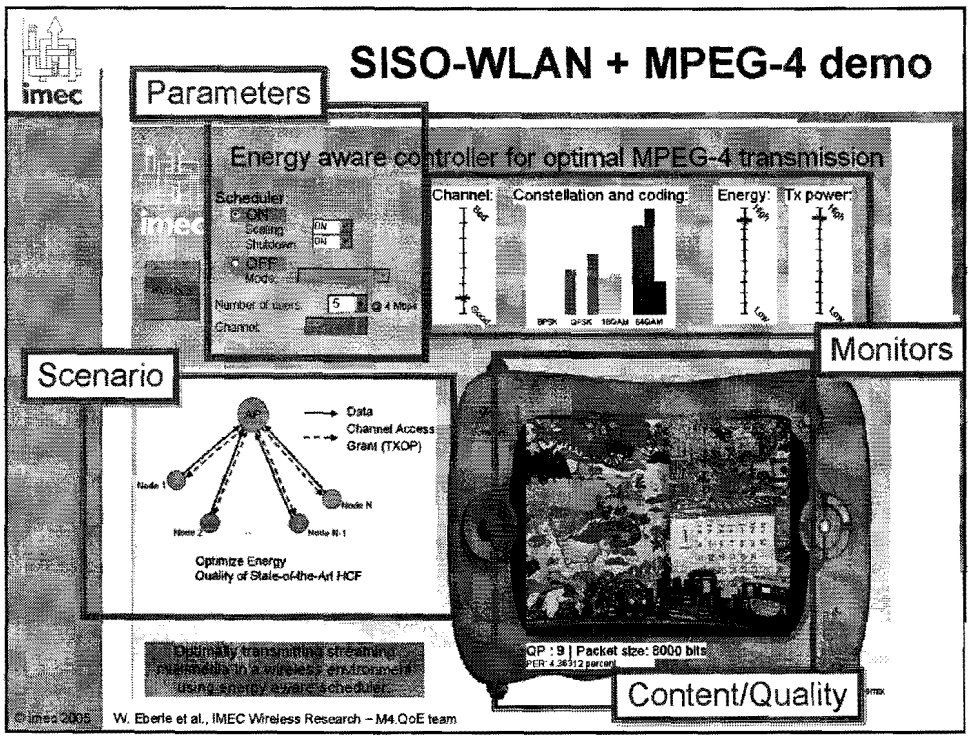


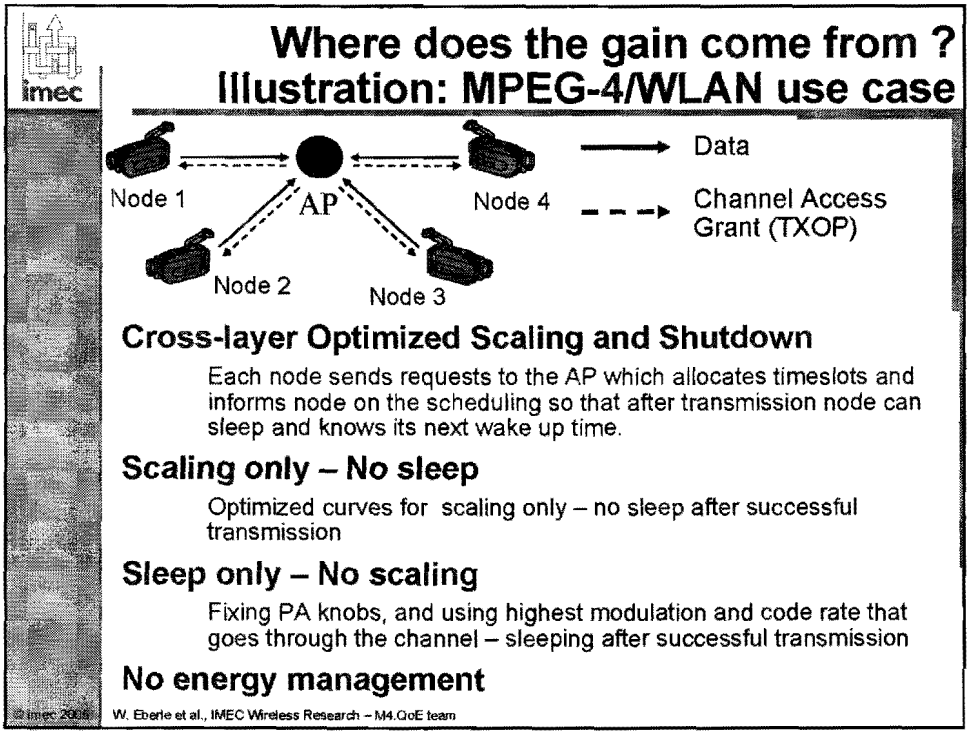
Examples

- Single-user SISO WLAN + MPEG-4
- Multi-user SISO WLAN + MPEG-4
- Extension to MIMO WLAN
- Extension to scalable MPEG-4
- Further extensions for multimedia

So what can we achieve? We can trade-off goodput vs. energy




SISO-WLAN + MPEG-4 demo
Parameters
Energy aware controller for optimal MPEG-4 transmission
Channel:
Constellation and coding:
Energy:
Tx power:
Monitors
Scenario
Data
Channel Access Grant (TXOP)
Optimize Energy
Quality of State-of-the-Art HCF
QP : 9 | Packet size: 8000 bits
Content/Quality
W. Eberle et al., IMEC Wireless Research – M4.QoE team

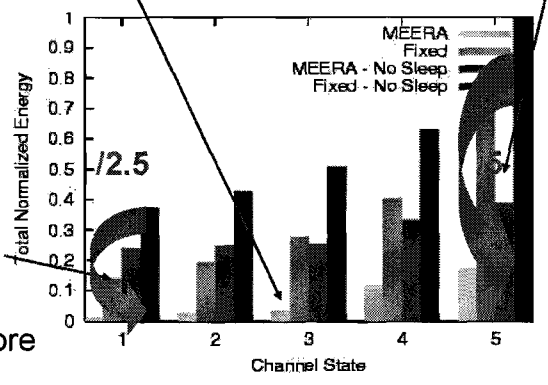

Where does the gain come from ?
Illustration: MPEG-4/WLAN use case
Node 1
AP
Node 4
Node 2
Node 3
Data
Channel Access Grant (TXOP)
Cross-layer Optimized Scaling and Shutdown
Each node sends requests to the AP which allocates timeslots and informs node on the scheduling so that after transmission node can sleep and knows its next wake up time.
Scaling only – No sleep
Optimized curves for scaling only – no sleep after successful transmission
Sleep only – No scaling
Fixing PA knobs, and using highest modulation and code rate that goes through the channel – sleeping after successful transmission
No energy management
W. Eberle et al., IMEC Wireless Research – M4.QoE team

Optimal cross-layer policy adapts to the circumstances

Optimal policy adapts and saves maximally exploiting multi-user diversity

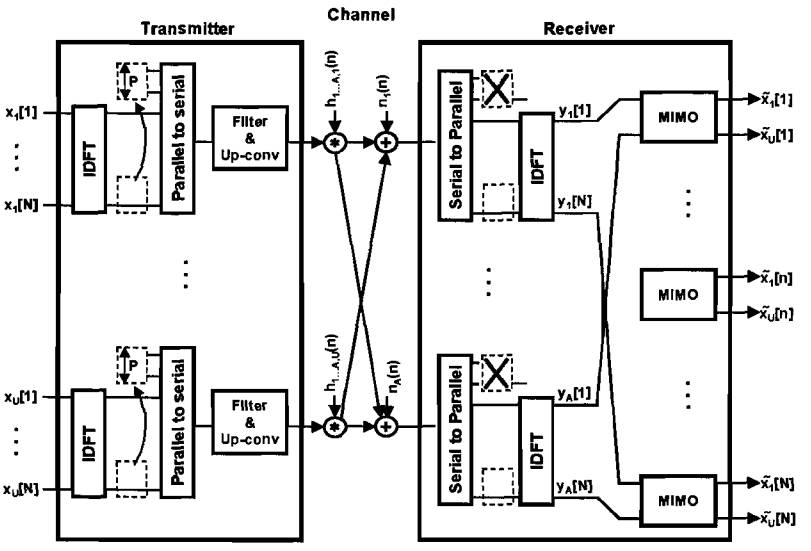
For bad channel: scaling saves more energy

For good channel: sleeping saves more energy



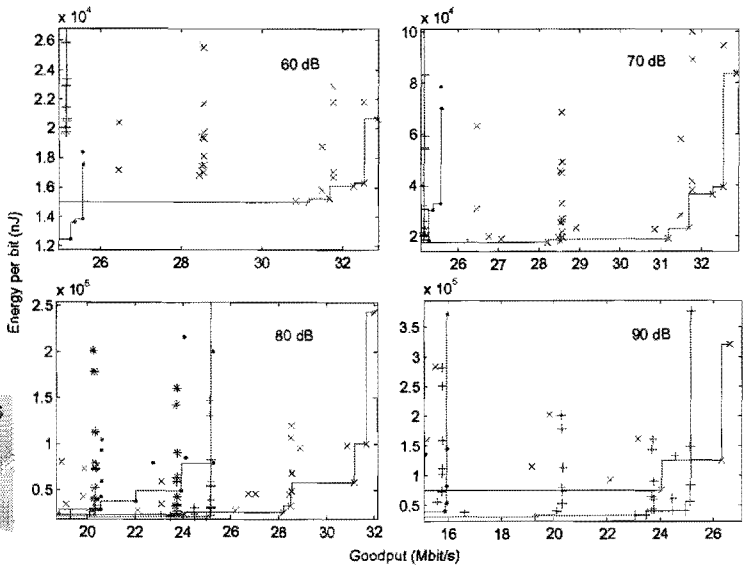
© imec 2005
W. Eberle et al., IMEC Wireless Research – M4.QoE team

Enhancing the wireless link From SISO to MIMO: STBC & SDM



© imec 2005
W. Eberle et al., IMEC Wireless Research – M4.QoE team

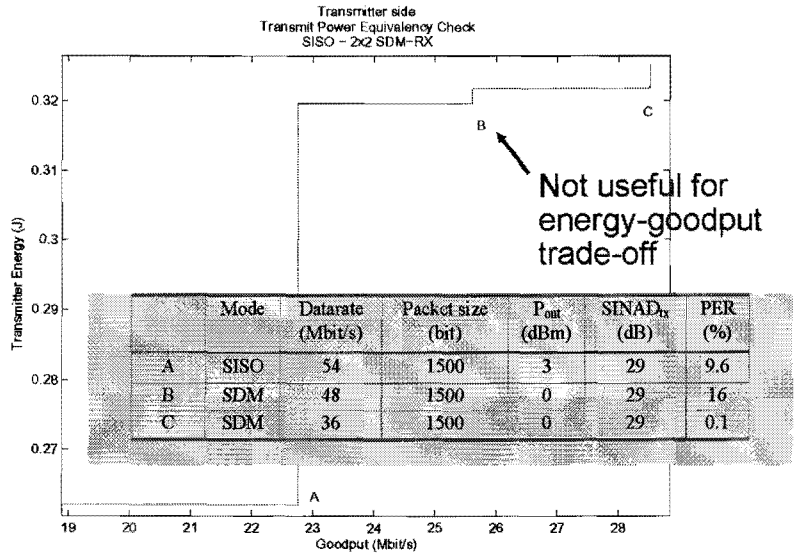
MIMO offers significant trade-offs between energy and goodput (path loss: 60,70,80,90dB)



MIMO modes
( ) SISO
(+) STBC
(x) SDM

W. Eberle et al., IMEC Wireless Research – M4.QoE team

When to use which mode? Are all configurations useful?



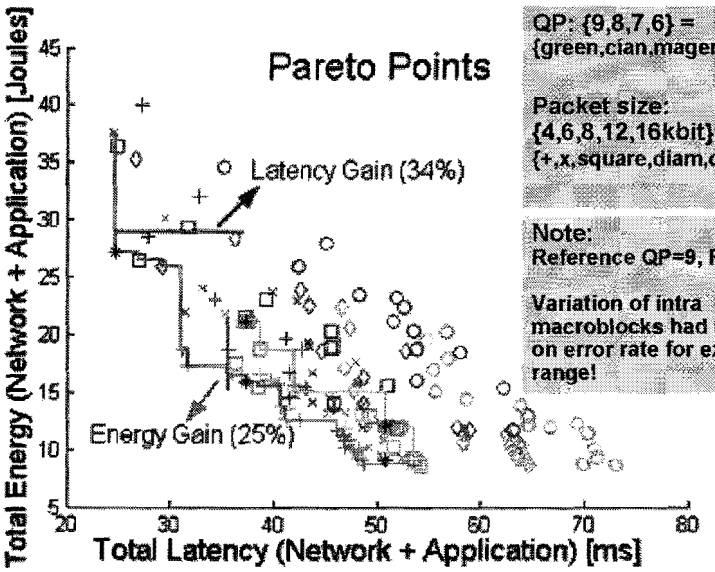
	Mode	Datarate (Mbit/s)	Packet size (bit)	P_{out} (dBm)	$SINAD_{rx}$ (dB)	PER (%)
A	SISO	54	1500	3	29	9.6
B	SDM	48	1500	0	29	16
C	SDM	36	1500	0	29	0.1

W. Eberle et al., IMEC Wireless Research – M4.QoE team

MPEG-4 exploration so far

- No on-line adaptation of multimedia knobs to variations of channel conditions of network load.
  - Encoding parameters fixed for the whole sequence
- Energy - latency tradeoffs increased
  - For fixed latency, tuning of coding parameters can provide energy gain
  - For fixed energy, tuning of coding parameters provides latency gain

Even MPEG-4 SP offers already some energy-latency trade-off range



QP: {9,8,7,6} = {green,cian,magenta,blue}

Packet size: {4,6,8,12,16kbit} = {+,x,square,diam,circ}

Note:
Reference QP=9, PS=8kbit

Variation of intra macroblocks had no effect on error rate for explored range!

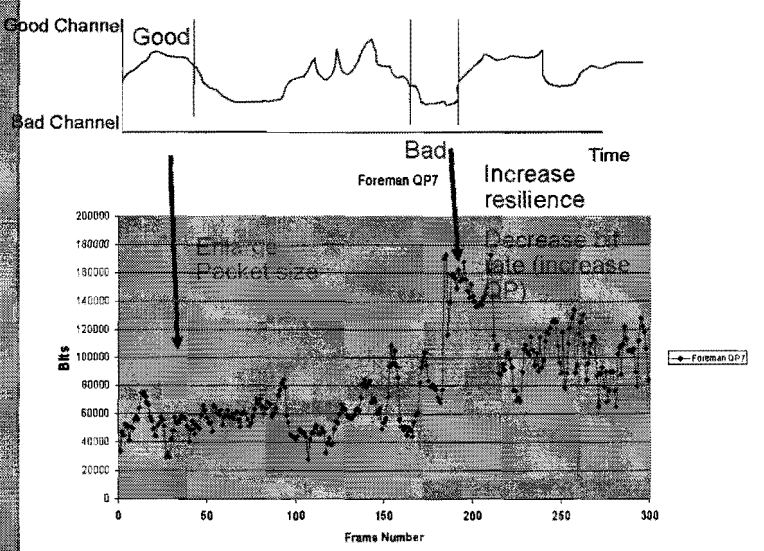
MPEG-4 Adaptability to Network Variations

- On-line adaptation to channel variations
  - Channel coherence time ~ 100ms
  - Encoding at 30 frames per second:

  Adaptation of coding parameters every 3, 4 video frames
- Quantization Parameter (QP)
  - determines the accuracy of representing the information
  - impacts the error-free quality, output bitrate
- Video packet size
  - coincides with MAC packet size
  - main impact on network energy
- Percentage of Intra MacroBlocks
  - increases robustness to errors at the cost of an increased bit rate

© imec 2005

Video Adaptation

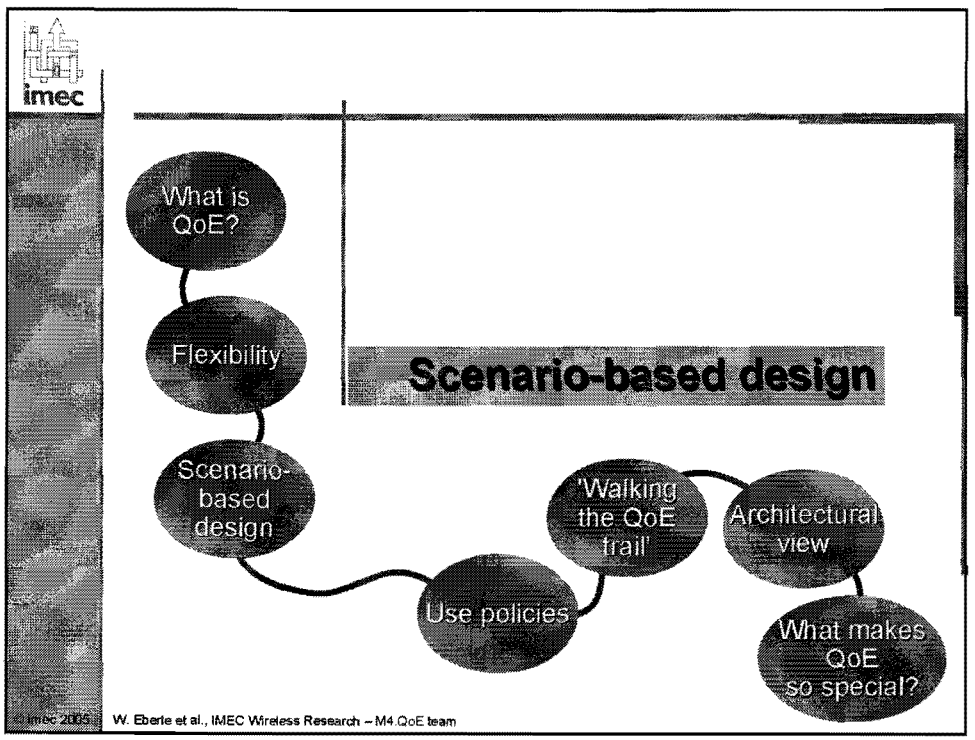


© imec 2005

More advanced codecs

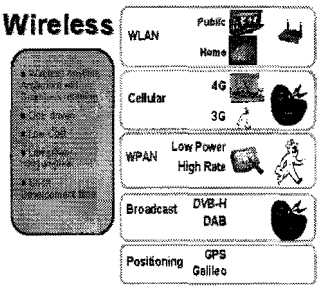
- Advanced Video Codec (AVC, MPEG-4 Part 10)
  - Enhanced coding efficiency
  - Error resilience tools:
    - Flexible Macroblock Ordering (FMO) for burst errors
    - Arbitrary Slice Order
    - Data Partition (Different kind of info separated)

    + Unequal Error Protection:
      - Queue priorities in scheduler
      - Less energy spent on less important information
- Scalable Video Codec (AVC based)
  - Fine grain scalability achieved
  - How to protect each layer for maximizing quality?
    - Error Resilience + FEC, ARQ
    - Impact of every layer on end quality?
    - Prioritization of different information at scheduler?



Scenarios – use cases

Scenario
Use cases
Standards
Requirements



Multimedia

MPEG-4 H.264(AVC)
DivX SVC

The more flexibility, the more overhead cost ...

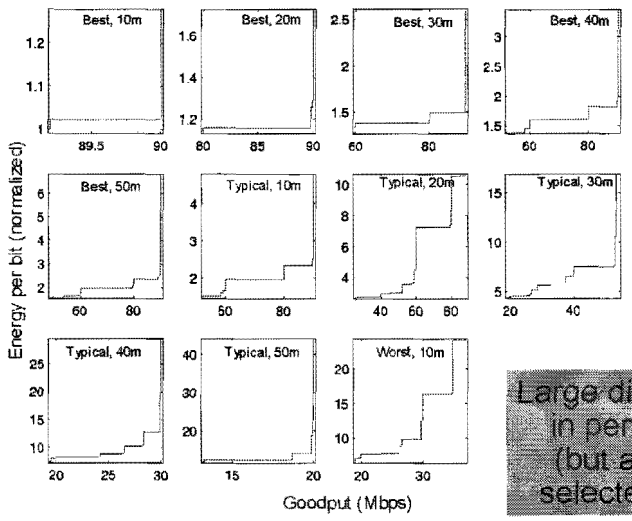
Covering extreme cases lead to extreme cost ...

Know for what you design the system → use cases:

- QoS requirements
- (multi-user) traffic patterns
- wireless environment
- service types
- options: multimedia, wireless

© imec 2005

What can happen?



Large differences in performance (but also in the selected knobs)

© imec 2005

Scenario-based design is not new

- Significant use e.g. in software design
  - Reference: M. B, Rosson, J. M. Carroll, "Scenario-based design," in Chapter 53, J. Jacko, A. Sears (Eds.) *The human-computer interaction handbook: Fundamentals, Evolving Technologies and Emerging Applications*, Lawrence Erlbaum Associates, 2002, pp. 1032-1050.
- What is it?
  - Family of techniques in which the use of a future system is concretely described at an early point in the development process.
  - Advantage: relatively lightweight (not a formal technique)

Scenarios in M4: flexibility vs. cost

- Impact of scenarios
  - For a system with a very narrow focus (e.g. wireless voice communication only):
    - wrong scenario defined => wrong product
    - right scenario defined => right product, maybe too costly since designed for the extremes
  - For a system with a very wide focus such in M4
    - what are the relevant scenarios?
    - what will the user do?
    - we can not design for all extremes (e.g. for the 1% of the use cases) for the same cost, energy, performance, etc.
- Conclusion
  - Scenario definition is crucial to design the right product
  - Coverage of the scenario space is crucial to design the right product with the right properties (e.g. flexible/adaptive in the right way)

Motivation for scenario-based design

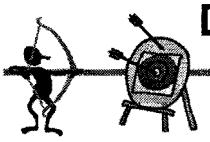
Potential gain/benefit of a particular knob depends on whether

- the knob is used often enough;
- the knob is unique (= non-replacable);
- the knob can be efficiently steered

Hence, need for scenarios to make sure

- we treat relevant practical cases => statistical weight for extreme cases/edges
- we have a true multi-mode multi-media system with sufficient externally determined behavior (user, environment) => testbenches
- we have specs based on which we judge the system results => specs, metrics

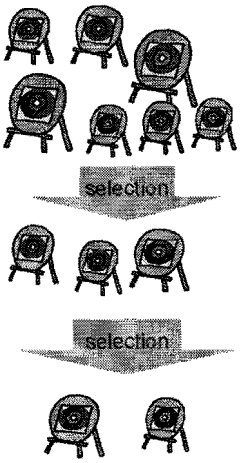
Different scenarios sets in M4



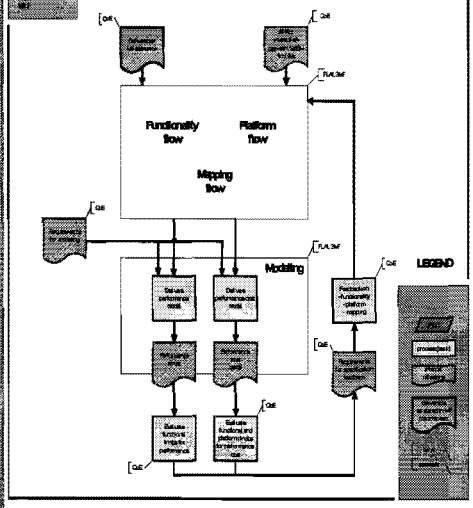
The M4 terminal

Need to distinguish:

- scenarios for exploration
  - goal: determine performance cost, compare to known cases
    - includes extreme cases
    - may include non-relevant cases
    - includes reference cases
- scenarios for design
  - goal: cover all relevant cases
  - reference/test cases for
    - system design
    - deriving component specs
- scenarios for real-time prototype demonstrators
  - goal: proof-of-concept of the particularly innovative features, reasonably low effort (!)

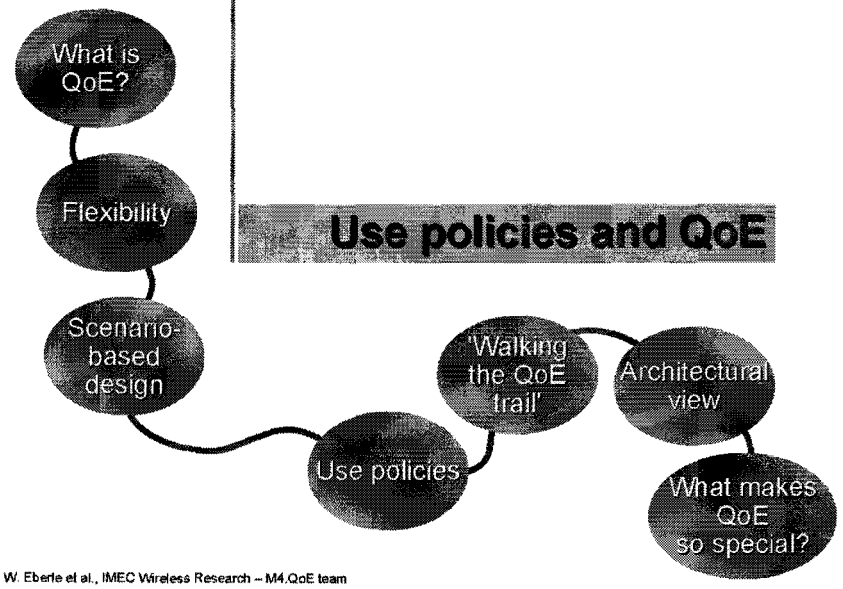


Example for the role of QoE: Interaction QoE and component



- QoE as server
  - provide scenario spec
  - provide modeling spec
- QoE as client
  - receive func spec
  - receive perf and energy-perf models (multiple times ...)
- QoE as server
  - provide evaluation results (energy-perf, run-time ctrl)
  - provide p-perf gain/benefit (for the overall system) = quality of the solution

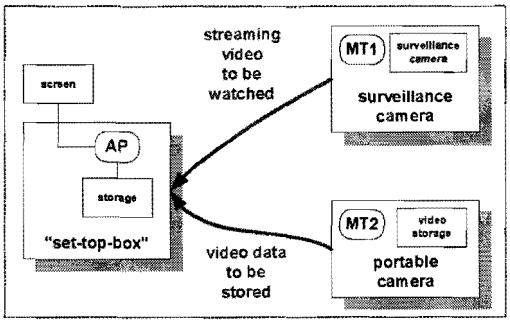
W. Eberle et al., IMEC Wireless Research – M4.QoE team



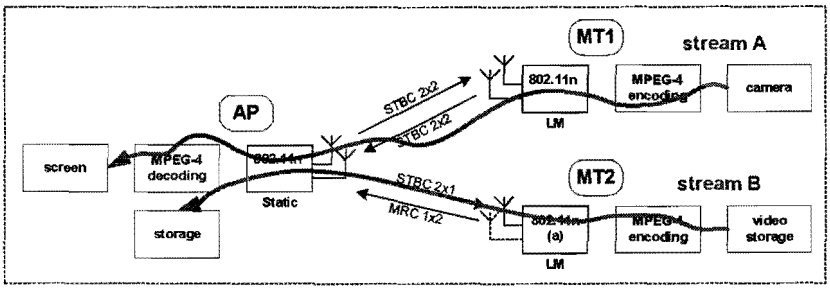
W. Eberle et al., IMEC Wireless Research – M4.QoE team

Scenario for the first M4 integrated demo (preliminary)

Nigel and Tamara live in a house with garage and a large garden. For security reasons, they have surveillance cameras installed. They find it convenient that they can watch the camera stream from different screens in the house, e.g. from their TV screen. Nigel and Tamara are also fond of taking videos with their camera. They have a collection of videos stored on their multimedia PC; so, regularly they upload the videos stored in the camera onto the PC.



Scenario in a component view (designer's perspective)



Same scenarios expressed as use cases (users's perspective)

- The camera surveillance consumes a lot of power when streaming; Nigel would like to know for **how long** he can still watch the video stream from the surveillance camera before local power supply/batteries need to be replaced. He does not always need high quality video but he wants to be able to **increase quality on demand**.
- Tamara usually does not put the camera close to the PC when uploading stored video streams from the camera to the PC. She expects the **upload time to be as short as possible** but she does not have unexpected failures in case the battery is not charged enough.
- Sometimes, Nigel sets up the surveillance while Tamara is already uploading a video from the camera to the PC. The network should **allow both operations** but Nigel wants to **control the quality** of the surveillance stream.

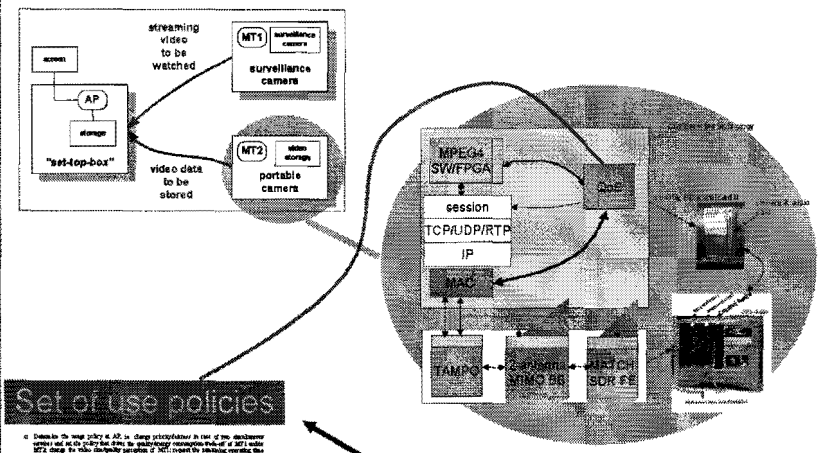
Common: (multi-objective) optimization
Different: role of constraint and target-of-optimization

Resulting use policies for the QoE manager

User expectations differ and require different policies:

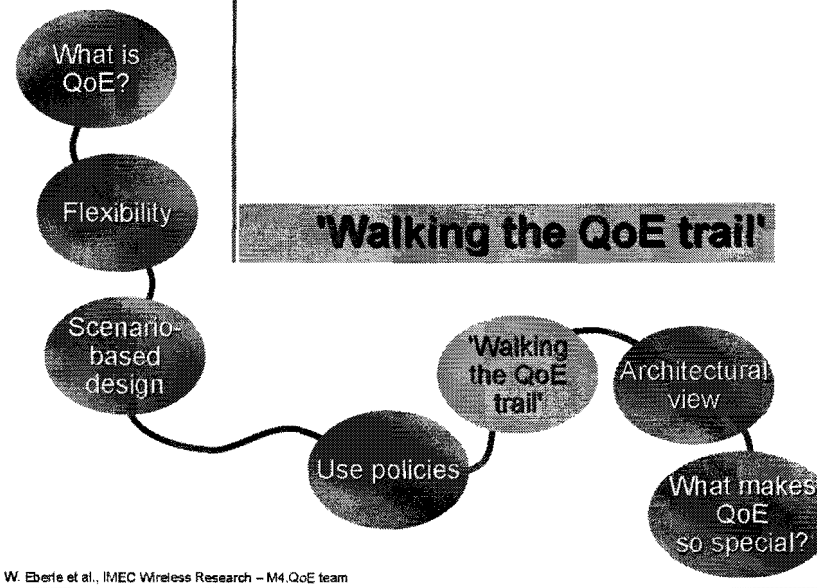
- maximum operating time for MT1 for video streaming with a user-determined level of quality;
- maximum quality for MT1 and MT2 for video streaming (give a prediction of time) at minimum energy cost;
- maximum upload speed (at a desired quality level)
- enforce highest priority for video stream originating from MT1 when receiving simultaneous streams from MT1 and MT2;
- adjustable level of fairness for both streams when received simultaneously (give a prediction of download time for MT2).

Possible implementation view for the real-time prototype



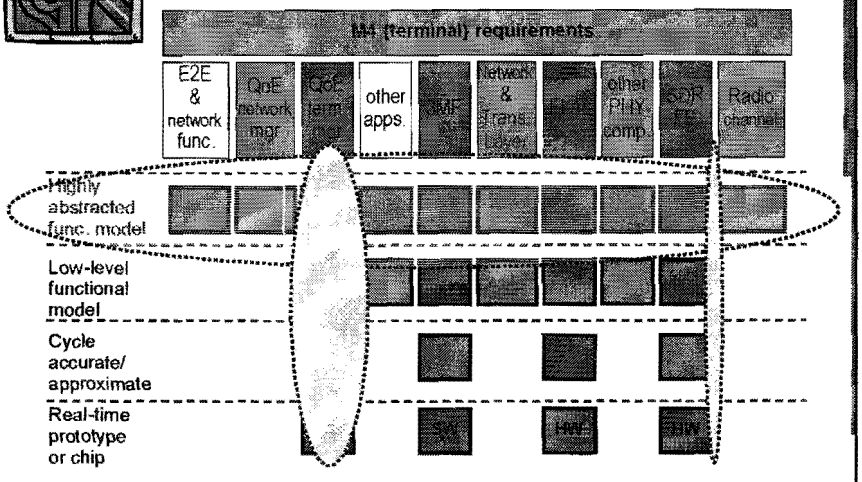
Set of use policies

This is where QoE brings user and system into contact!

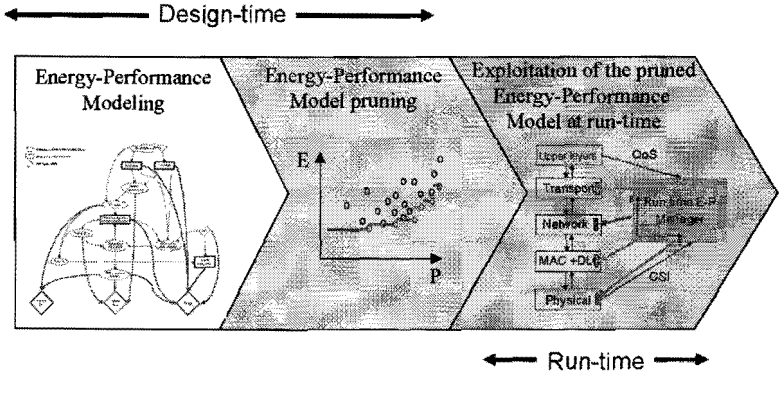


'Walking the QoE trail'

Where is QoE?



Splitting the work between design- and run-time



Flow at design-time and run-time

DESIGN TIME

- Scenario spec
  - Scenario description
  - Functionality definition (e.g. compliance to standards)
- Functional exploration
  - Define pool of algorithms
  - Perf-quality modeling
  - *Exploration*
- Func/Architectural exploration
  - Define arch/platform options
  - Mapping alg > platform
  - Perf-quality-cost modeling
  - *Exploration*
- Provide final spec

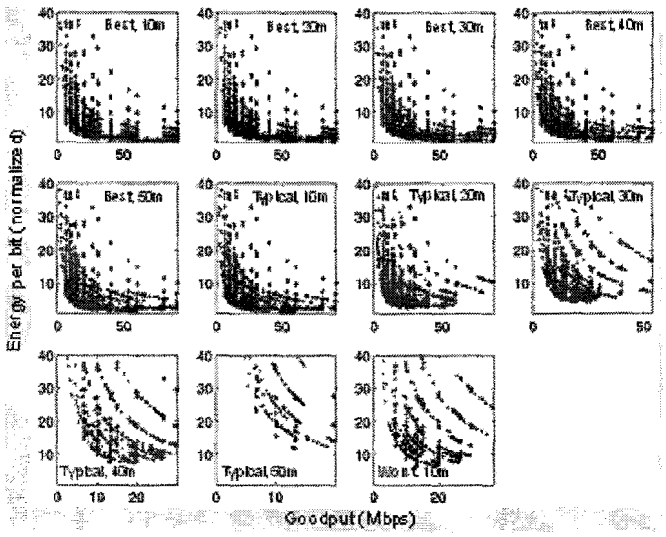
Exploration at design-time

- Merge individual characteristics
- Determine irrelevant knobs
- Prune granularity of knobs
- Resampling for compaction
- Result : can be still different Pareto curves for individual parts of the system or for independent users/services/streams etc.

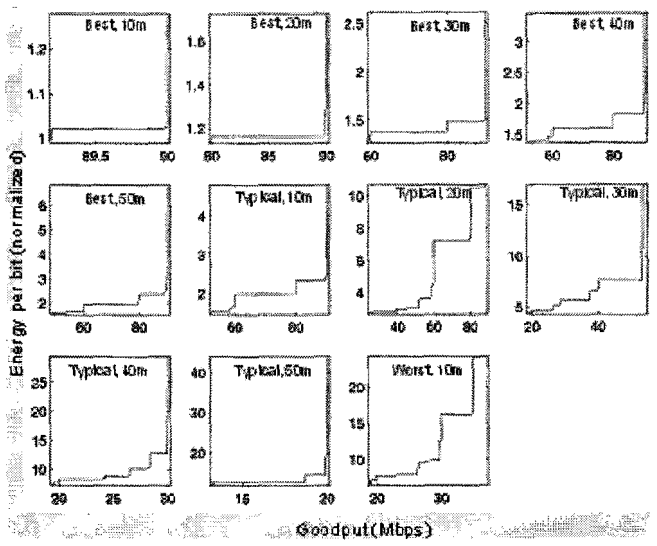
RUN-TIME

- Estimate system + environment state
- Pick appropriate Pareto curves
- Merge Pareto curves (merging)
- Find optimum system configuration (searching)
- Configure the individual subsystems

Exploration results before pruning (Pareto clouds)

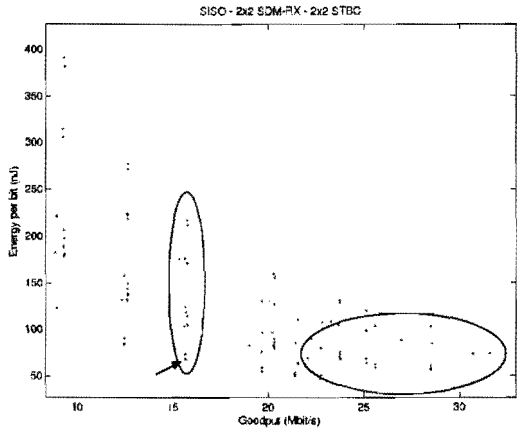


Exploration results after simple pruning (Pareto front)

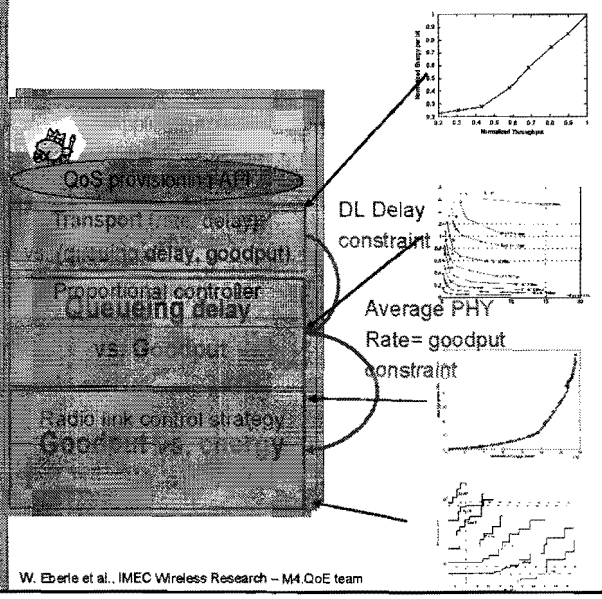


Deriving the Pareto-front

- Pareto front is to the right: energy increases with goodput
- Points with higher energy for the same goodput can be pruned



We can partition the global optimization problem into steps!



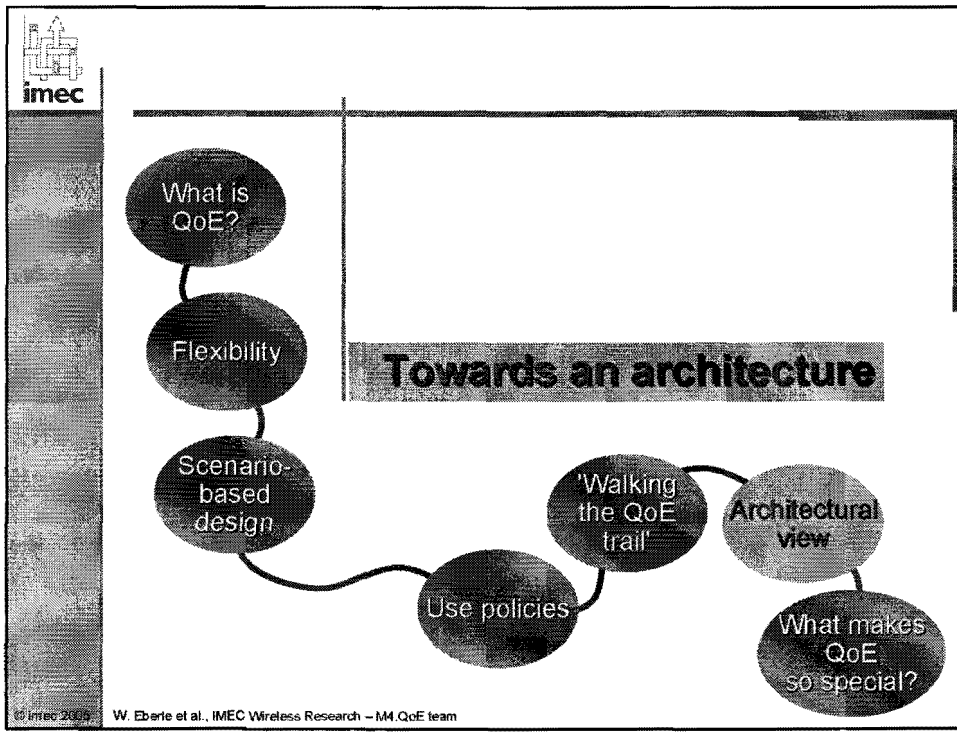
TCP throughput vs. energy/bit trade-off

DL average delay vs average energy/bit trade-off

PHY average rate vs. energy /bit tradeoff (run-time)

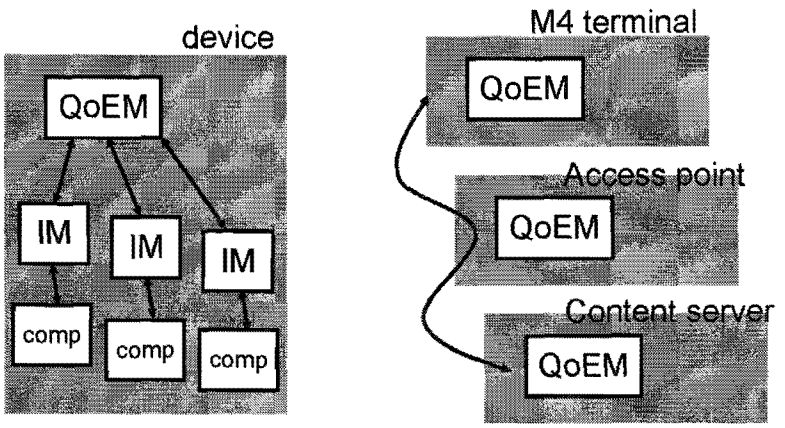
PHY layer rate vs. energy trade-off curves (Design-time)

© imec 2005

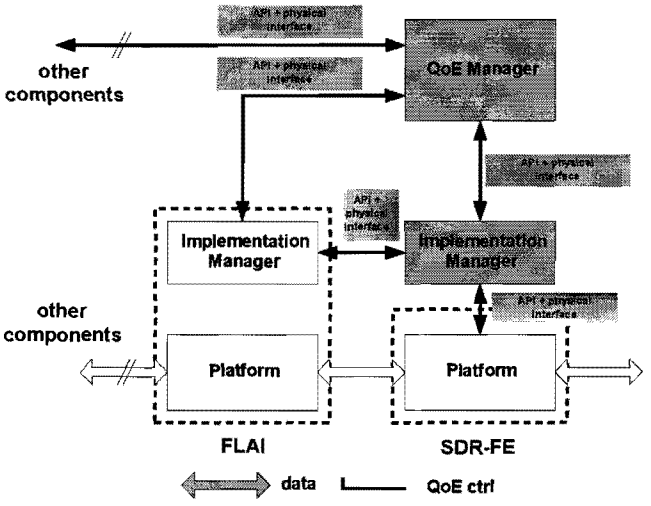


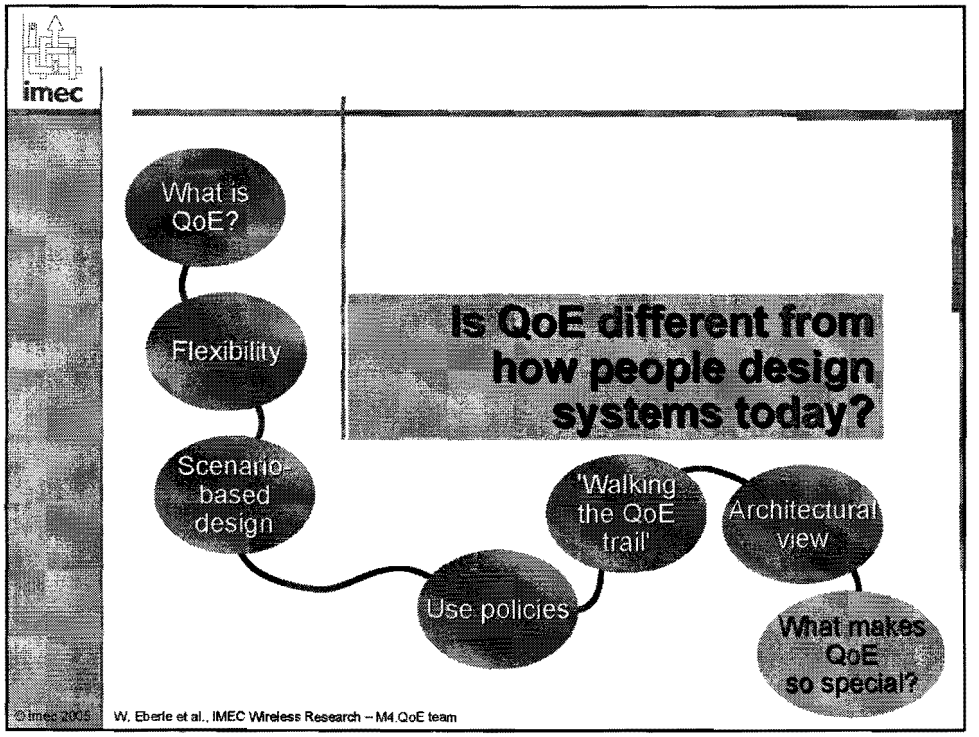
© imec 2005

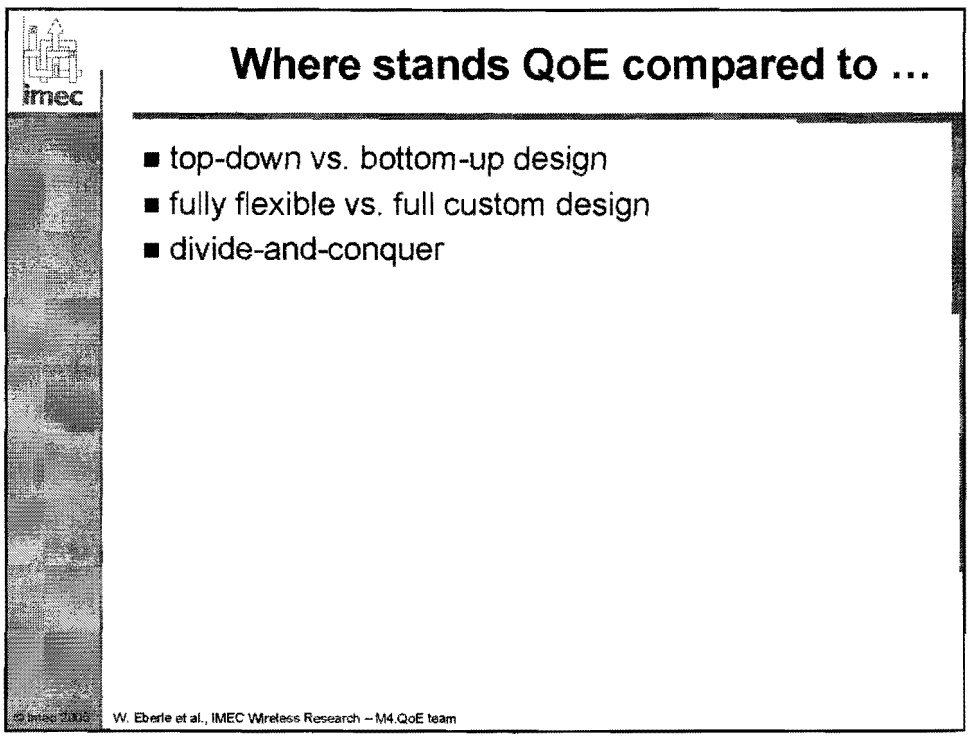
QoE management works in a hierarchical way



QoE architectural view



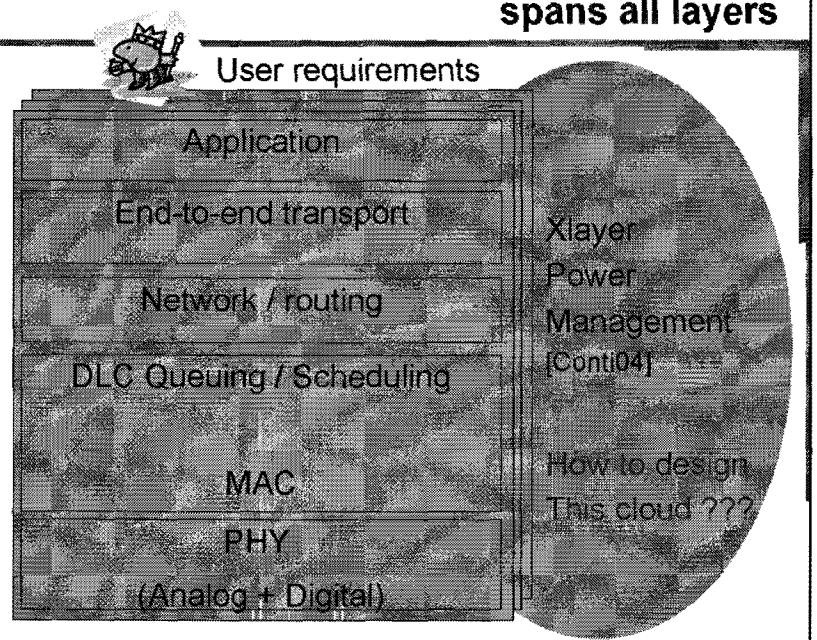

imec
What is QoE?
Flexibility
Scenario-based design
Use policies
'Walking the QoE trail'
Architectural view
What makes QoE so special?
Is QoE different from how people design systems today?
© imec 2005
W. Eberle et al., IMEC Wireless Research – M4.QoE team


imec
Where stands QoE compared to ...
top-down vs. bottom-up design
fully flexible vs. full custom design
divide-and-conquer
© imec 2005
W. Eberle et al., IMEC Wireless Research – M4.QoE team

Design approaches: up or down

- QoE is about system design
- The two extremes in system design:
  - Top-down design
    - Advantage: Global optimization possible
    - Disadvantage: maybe costly redesign of many IP blocks
  - Bottom-up, e.g. platform-based design
    - Advantage: based on actual (well-known) IP
    - Disadvantage: risk for suboptimal results at the system level (availability of the right IP, specifications for the IP, ...)
- In practice
  - Meet-in-the-middle = a bit of top-down and a bit of bottom-up
  - Problems:
    - how much is a bit?
    - where is the methodology here, how to repeat it?
    - how to ensure optimality?
    - how to ensure integrity?

QoE: Effective Power Management spans all layers



Design approaches: flexible vs. fixed

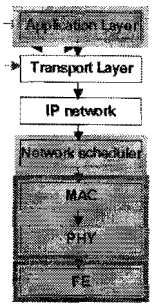
- QoE is about system design
- The two extremes in system design:
  - As flexible as possible (=> software)
    - Advantage: easy to upgrade/update
    - Disadvantage: energy-hungry; some problems are shifted to other domains (e.g. software radio: digital → analog)
  - As energy-efficient as possible (=> custom hardware)
    - Advantage: energy-efficiency
    - Disadvantage: fixed; long design time; redesign cost; manufacturing cost
- In practice
  - Meet-in-the-middle = a bit of SW and HW
  - Problems:
    - how much is a bit?
    - where is the methodology here, how to repeat it?
    - how to ensure optimality?
    - how to ensure integrity?

QoE goes for the real trade-off between flexibility and energy

- Extensive exploration phase @ design-time
  - Advantage: optimum configurations can be determined; not-required parameter flexibility is identified early (before detailed design starts); energy is traded off against flexibility!
  - Disadvantage: models must be available (note: this is the same problem as for deriving specifications; hence: nothing new!)
- Pruning of flexibility to the optimum @ design-time
  - Advantage: minimum complexity remains for run-time
  - Disadvantage: none (methods & tools needed, of course)
- Interactive control (monitor & steer) @ run-time
  - Advantage: can still react to changing environment, services, etc.
  - Disadvantage: some overhead in (mainly) digital complexity; however: negligible compared to savings

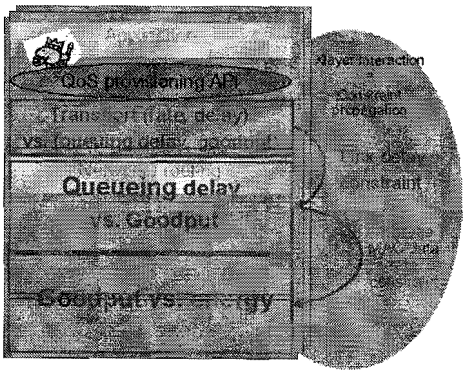
Design approaches: divide & conquer

- Divide-and-conquer
  - Advantage: good engineering principle as such; reduce complexity of the original problem by splitting it up into subproblems of lower complexity while increasing their number
  - Disadvantage: often misused – why?
    - What is the design criterion to split a problem?
    - Where to partition problems?

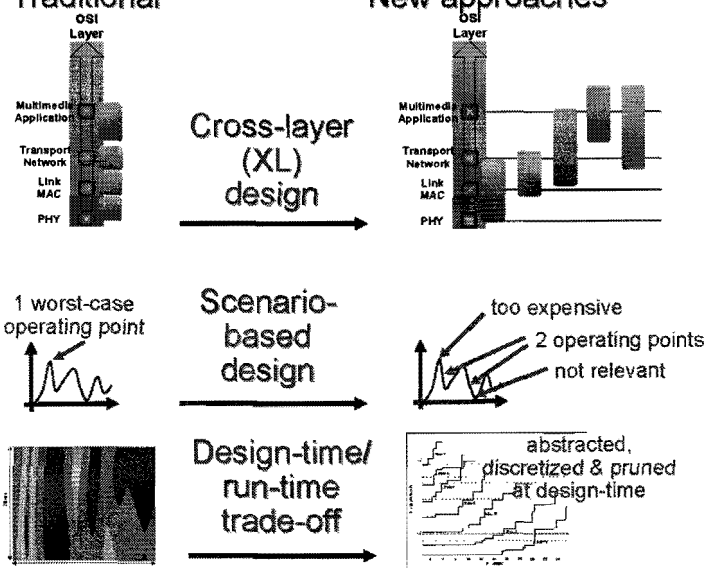


QoE is not about chaos but about minimizing the interfacing cost

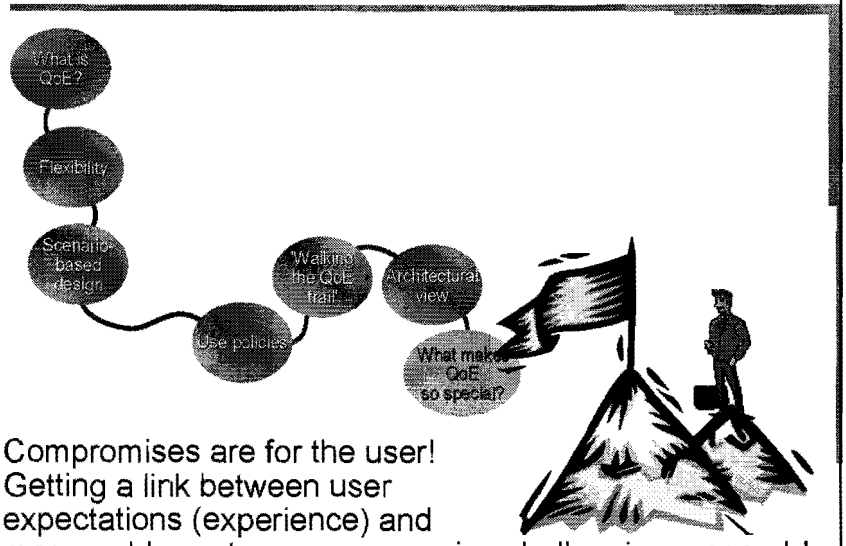
- Layering is still possible!
- Instead: Optimize interfacing cost
  - QoE identifies the additional *or* the actually needed (often irrelevant information is supposed to be exchanged) information that needs to be exchanged



Can we improve our design approach?



You can't have it all at the same time ... but you will get the 'best' possible at any given time



Compromises are for the user!
Getting a link between user expectations (experience) and measurable system specs remains challenging research!

Short Papers

A Characterization of Streaming Applications Execution

M.A. Weffers-Albu[1], J.J. Lukkien[1], P.D.V. v.d. Stok[1,2]

[1]Technische Universiteit Eindhoven, [2]Philips Research Laboratories

Abstract

In this article we provide a model for the dynamic behavior of a single video streaming chain, by formulating a theorem describing the stable behavior. This stable behavior is characterized in terms of the elementary actions of the components in the chain, from which standard performance measures (such as activation time, response time and resource utilization) follow. From this we derive corollaries, which give guidelines for the design of the chain, while targeting optimization of resource utilization and minimizing context-switching overhead.

1. Introduction

We consider the problem of processing a video stream by an application consisting of a chain of given processing components, on a scarce-resource embedded platform. The essential requirement on the platform is cost-effectiveness, leading to minimizing the resources made available to this application. The requirement on the application is robustness. In terms of the real-time tasks that compose the application, this leads to the requirement of predictability of timing behavior and (shared) resource use. In order to support the design of the application (mainly the mapping of such an application onto an execution platform) a good model of its run-time behavior is needed such that timing and resource utilization can be accurately predicted. In addition, the model can be used to control that behavior.

The parameters we want to predict and control are attributes such as activation time (AT) and response time (RT) of tasks and of the chain as a whole, as well as resource utilization (RU) for CPU, memory and bus. Predicting and controlling the RT of each task in the chain is important for the prediction and control of the response time of the entire chain. This article presents a first step in this work by characterizing the execution of a single video streaming chain by formulating a stable-phase theorem. Subsequent corollaries provide guidelines for design aiming at improving the resource utilization and minimizing the context switching overhead. We adopt as an important measure for the overhead the number of context switches (NCS).

The article presents our execution model in section 2, and a characterization of a single streaming chain execution in section 3. Related work is presented in section 4 and section 5 is reserved for conclusions.

2. TriMedia Streaming Software Architecture

This work has been done in collaboration with the *Multi-Resources Management* project at Philips Research Laboratories Eindhoven where we study the TriMedia Streaming Software Architecture (TSSA). TSSA is an instantiation of the *pipes and filters architecture style* [6] and it provides a framework for the development of real time audio-video streaming applications executing on a single TriMedia processor [1]. A media processing application is described as a graph in which the nodes are software components that process data, and the edges are finite queues that transport the data stream in packets from one component to the next (Figure 1).

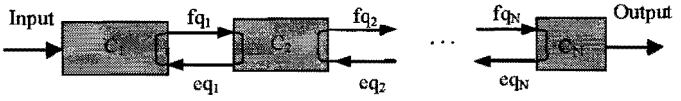


Figure 1 - Chain of components.

Each component C_i ($1 \leq i \leq N$) has an associated task to which a unique fixed priority is assigned (fixed priority scheduling) and the tasks execute as long as input is provided. Every connection between two components is implemented by two queues. One queue (called *full queue*) carries *full* packets containing the data to be sent from one component to the next, while the second queue (the *empty queue*) returns *empty* packets to the sender component to recycle packet memory. The empty packets are returned in order to signal that the data has been received properly and that the memory associated with the data packet may be reused. Each component C_i ($1 < i < N$) has two input queues (a full queue fq_{i-1} and an empty queue eq_i) and two output queues (a full fq_i and an empty queue eq_{i-1}) (Figure 1). C_1 and C_N are connected to their neighbors by only two queues. C_1 has one input empty

queue eq_1 and one output queue fq_1. C_N has one input full queue fq_{N-1} and one output queue eq_{N-1}.

A typical *execution scenario* of C_i (denoted by $E(C_i)$) (Figure 2) is the following: the component gets 1 full packet from the input full queue (fq_{i-1}?) ❶, then gets 1 empty packet from the input empty queue (eq_i?) ❷, performs the processing (α) ❸, recycles the input packet by putting it in the output empty queue (eq_{i-1}!)❹ and, finally, the result of processing is put in the output full queue (fq_i!) ❺.

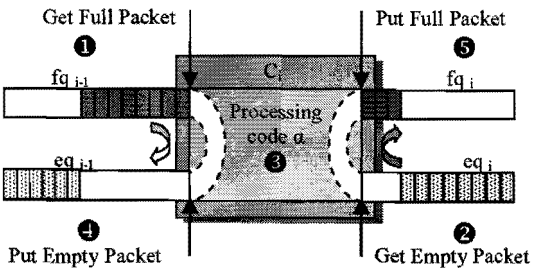


Figure 2. A basic streaming component.

Note that the semantics of the fq_{i-1}? and eq_i? operations is that the number of packets in the fq_{i-1} and respectively eq_i queue is decreased by one. Also, the semantics of the eq_{i-1}! and fq_i! operations is that the number of packets in the eq_{i-1}! and respectively fq_i! queue is increased by one.

3. A characterization of chain execution

In the current article we will focus on the case of a single linear streaming chain (Figure 1) consisting of event-driven components with fixed worst-case execution times and executing in a cooperative environment. The latter means that the environment will always provide input and always accept output. In general, a chain will not exist in isolation, but composed with other chains, the components can have different execution scenarios, and the environment is not always cooperative. As we mentioned in the previous section, the analysis we present in this paper is a first step in analyzing this type of streaming systems, step we consider necessary before tackling more complex cases. We also observe that the (environment) input and output of the chain are different than the connections (via buffers) between components because while C_1 and C_N will never block on the environment input and respectively output, they can block on their input empty and respectively full queue.

The initial situation of the chain is that all full queues are drained (which implies that all empty queues are filled to their full capacity). At any time the task with the highest priority that has enough input to run will execute. We will use the following notations: P_i refers to the priority of component C_i, L(Q) denotes the number of packets in queue Q, |Q| denotes the capacity of queue Q, C_i **b** Q means that component C_i is blocked on queue Q. C_M denotes the component with minimum priority in the chain.

In order to characterize the system behavior we list invariant properties that we derive from the components behavior and from the priority assignment. Assuming the capacities of empty queues are identical to the capacities of the corresponding full queues we derive from the components behavior:

Property 1 – $\forall$ i, $1 < i \leq N$, C_i **b** eq_{i-1} is not possible.

Property 2 –$\forall$ i, $1 \leq i < N$, C_i **b** fq_i is not possible.

Property 1 and 2 state that blocking at the output of a component is not possible.

Whenever a component executes, it can do so only if higher priority components are blocked. Consider a descending chain of priorities starting from the input side of the chain. Whenever the last component in this chain executes, the other ones are blocked. Because of the cooperative environment all components (except the last one) are blocked on reading from the empty queue. This situation can be generalized to a minimal priority component.

Lemma 3 - When C_i is such that $\forall$ j, $j<i$, $P_j > P_i$ and C_i is executing in α, then C_j **b** eq_j.

A similar result holds for the end of the chain.

Lemma 4 - When C_i is such that $\forall$ j, $j > i$, $P_j > P_i$, and C_i is executing in α, then C_j **b** fq_{j-1}.

For component C_M the situation is special in that both lemmas 3 and 4 apply. As a result, whenever C_M executes the remainder of the chain is blocked. In addition, whenever C_M de-blocks one of its neighboring components, the components in the corresponding half-chain will take over, execute one time their execution scenario after which they will return to their blocked state again. The sequence of actions in doing this is completely determined by the priority assignment. Therefore, the behavior of the system can be described as the interleaving of the behavior of C_M with these left and right half-chain behaviors.

Corollary 5 - When C_M is in α, $\forall$ i: $1 \leq i < M$, $L(fq_i) = |fq_i|-1 \wedge L(eq_i)=0 \wedge \forall$ i: $M \leq i < N$, $L(fq_i) = 0 \wedge L(eq_i)=|eq_i|$.

Stable Phase Theorem - Let $C_1, C_2, C_3, \ldots, C_N$ be a chain of event-driven components communicating through a set of queues as in Figure 1. Provided that the input is sufficiently long, the execution of the components in the chain will adopt a repetitive pattern (during which the chain is in a stable phase) in a finite number of steps (initialization phase). The repetitive pattern of execution is:

fq_{M-1}?; eq_M?; α; eq_{M-1}!; $E(S_L)$; fq_M!; $E(S_R)$,

where $S_L=\{C_1, \ldots, C_{M-1}\}$, $S_R=\{C_{M+1}, \ldots, C_N\}$ and $E(S_L)$ and $E(S_R)$ are the mentioned combined executions of components in sub-chain S_L and sub-chain S_R respectively. Note that the repetitive pattern of execution depends on the execution scenario of the components. In

future work we will consider chains composed of components with different execution scenarios and we will show again how the different execution scenarios influence the pattern of execution.

Property 6 - The length of the initialization phase in number of execution steps is $\sum_{j=1}^{M-1}\sum_{i=j}^{M-1} |\, eq_i \,|$.

Corollary 7 - The length of the initialization phase can be reduced by reducing the length of all queues connecting the components preceding C_M in the chain to the limit necessary to prevent deadlock.

Corollary 8 - The minimum queue length sufficient for each of the queues in the chain is 1.

Corollary 9 - The initialization phase can be eliminated completely by assigning to C_1 the minimum priority.

Corollary 10 – In the stable phase the execution of all components is driven by the execution of C_M.

Corollary 11 - At the beginning of the stable phase $\forall\, i \neq M$, C_i is blocked while C_M is ready-to-run.

Corollary 12 – NCS, AT, RT of all tasks involved, RT for the entire chain, and CPU utilization during one execution of the pattern can be calculated by reconstructing the first execution of the pattern considering that in the beginning of the stable phase the states of all component tasks are as described in Corollary 11.

The algorithm for reconstructing the execution of the pattern is described in [2].

The following corollary provides guidelines on how to assign priorities to tasks in the chain such that the NCS is minimized. We are interested in minimizing the NCS because we want to minimize the overhead.

Corollary 13

1. The minimum NCS during initialization phase can be achieved when $C_M = C_1$.
2. The minimum NCS during one execution of the pattern can be achieved either when:
   a. $P_1 < P_N < P_{N-1} < \ldots < P_2$,
   b. or when $P_N < P_1 < P_2 < \ldots < P_{N-1}$.

4. Related work

Closely related work in [3] and [4] also considers an execution model for video streaming chains inspired by TSSA. Both articles present an analysis method allowing the calculation of the worst-case RT of multiple video streaming chains based on the canonical form[1] of the chains. The assumptions adopted are that tasks have fixed execution times, tasks are allowed to have equal priorities and the overhead introduced by context switches is ignored. In contrast, in our analysis we can deal with variable execution times and take the overhead of context switching into account. In addition, we provide guidelines for the optimization of RU.

The execution model used in [3] and [4] presents only one buffer linking two consecutive components in a chain. Although we expect that a similar stable phase theorem could be stated also for this execution model, the corollaries derived would differ because of the different execution model used.

Finally, in [5] the authors focus on the fixed priority scheduling of periodic tasks decomposed into serially executed sub-tasks but no intermediate buffers are considered.

[1] The canonical form of a chain CN composed of N tasks, can be seen as a new chain CN' composed of N' tasks ($N' \leq N$) where the priorities of the tasks do not decrease from input to output.

5. Conclusions

We have presented a characterization of the dynamic behavior of a video streaming chain composed of event-driven components. The presented theorem and lemmas show that after a finite initialization phase streaming chains reach a repetitive pattern of execution. This repetitive pattern is exploited further in predicting attributes such as activation time, response time of individual tasks, response time of the entire chain, the number of context switches and resource utilization. Additional corollaries provide guidelines for the design, while targeting optimization of resources and minimizing context-switching overhead.

Acknowledgements

We are thankful to Reinder Bril, Liesbeth Steffens, Clara Otero-Perez, Laurentiu Papalau, Giel van Doren and Dietwig Lowet for their helpful comments and suggestions during the process of reaching the present results.

6. References

[1] P.N. Glaskowski. Philips advances TriMedia architecture – New CPU64 core aimed at digital video market. *Microdesign Resources, 1998, vol.12, no 14, pp. 33-35.*

[2] M.A. Weffers-Albu, P.D.V. v.d. Stok, J.J. Lukkien. NCS Calculation Method for Streaming Applications. *Proceedings of the 5th PROGRESS workshop on embedded systems,2004,pp.3-9.*

[3] Angel M. Groba, Alejandro Alonso, Jose A. Rodriguez, Marisol Garcia-Valls. Response Time of Streaming Chains: Analysis and Results. *Proceedings of the 14th Euromicro Conference on Real-Time Systems, 2002, pp 182-192.*

[4] Angel M. Groba, Alejandro Alonso. Response Time Analysis of Periodic Chains. *The 22nd IEEE Real-Time Systems Symposium, 2001,Work in Progress, pp. 29-32.*

[5] Michael Gonzalez Harbour, Mark H. Klein, John P. Lehoczy. Fixed Priority Scheduling of Periodic Tasks with Varying Execution Priority. *Proceedings of the IEEE Real-Time Systems Symposium, 1991, pp.116-128.*

[6] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal, *Pattern-Oriented Software Architecture*, John Wiley & Sons Ltd., 1996.

Providing Adaptive QoS in Wireless Networks by Traffic Shaping

Tomas Lennvall, Gerhard Fohler
Deptartement of Computer Engineering
Mälardalen University, Västerås, Sweden
{tomas.lennvall,gerhard.fohler}@mdh.se

1 Overview

Our goal is to stream video over a wireless network by dynamically adapting the size of the stream according to the fluctuating available bandwidth in the network.

In this paper we propose an architecture that provides QoS by dynamically adapting the transmission rate of nodes to match the currently available bandwidth of a wireless network. The architecture prioritizes real-time traffic over non real-time traffic when transmitting packets. The traffic adaptation serves the purpose of minimizing the network congestion occurring due to high load.

The architecture consists of a *bandwidth predictor* that first uses a simple probe-packet technique to predict the available bandwidth of the network. Then, exponential averaging is used to predict the future available bandwidth based on the current measurement and the history of previous predictions.

This predicted bandwidth is then fed into the *traffic shaping* part of the architecture, which adjusts the transmission rate of the node accordingly.

As presented in [4, 2], the basic idea of applying traffic smoothing for network transmission is to smooth a bursty stream of data into a constant stream of data by using the leaky bucket algorithm. The rationale is that by smoothing out bursts the transmission is evenly spread out over time in order to reduce the probability of congestion and collisions on the network. It is also a way to control the rate of the transmitted traffic generated by each node.

As motivation for the work in this paper we use the two application scenarios described below.

The first application is a streaming server capable of dynamically switching between a small number of differently sized versions [5] of the same stream. Synchronization of the streams is based on GOPs, i.e. a switching between streams always occurs at a GOP start. The server switches the transmission between these streams according to the currently available bandwidth predicted by our method.

The second application is also a streaming server but it uses a web camera to capture pictures which are then encoded into a video stream with the possibility to control the stream size by adjusting a quality parameter of the encoding. The quality parameter is determined based on the bandwidth prediction we perform. This stream is then transmitted onto the network in a way so that it fits the available bandwidth.

2 Architecture

In this paper we assume all nodes are connected using the *Infrastructure* connection, i.e all nodes are connected to an *Access Point (AP)*. Furthermore we assume that there is only one AP in the system, hence no roaming between different APs can occur even though nodes are mobile. We assume that nodes are always within transmission and reception range of the AP.

The basic idea is that each node is assigned a proportional share of the total available bandwidth of the wireless network and each node stays within its limit. Here we assume that when the available bandwidth fluctuates, the proportional share of each node also fluctuates. We consider the node containing a streaming server most important, and therefore we assign it a larger share of the available bandwidth than the other nodes. Note that, because of the previously mentioned fluctuation of available bandwidth, the assigned share for each node will also fluctuate over time. This implies that the traffic generated by a node must be dynamically adapted to conform to the currently available bandwidth share for that node.

On each node the bandwidth assigned to that node is shared between real-time and non real-time traffic. Because we consider the video stream to be real-time, we assign a higher share of the allocated bandwidth to real-time traffic in order to prioritize the transmission of the video stream.

An overview of the architecture is presented in figure 1.

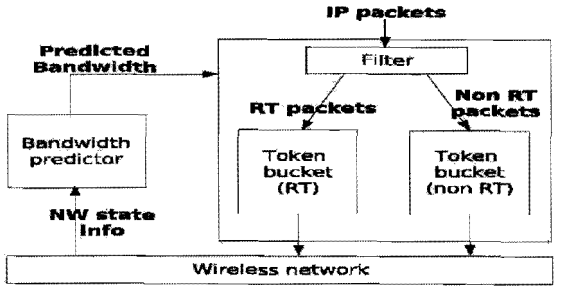


Figure 1. Architecture of bandwidth prediction and traffic shaping

In order to properly adjust the transmission rate, the first part of our architecture, the *bandwidth predictor*, predicts the future bandwidth based on the current available bandwidth and previous bandwidth predictions.

The bandwidth prediction is then fed into the second part of our architecture, the traffic shaper. It shapes the traffic according to the available network bandwidth. By separating the handling of real-time and non real-time traffic by using two token buckets (see Figure 1), we prioritize real-time traffic by giving it a larger share of the bandwidth assigned to the node.

Available bandwidth is predicted between one sender and one receiver, i.e if there are multiple receivers, available bandwidth must be predicted for each, because the results can be different depending of the fluctuations of the wireless network. In this paper we assume that each sender has only one receiver present.

3 Bandwidth Prediction

Our bandwidth prediction uses a well known *packet-pair* probing technique detailed in [3], which is a way to determine the end-to-end characteristics of a network path mainly for Internet, which we apply to wireless networks. The network state can be determined by measuring the time delay of the probe packets.

In order to dynamically react to the varying bandwidth of the network, we have to periodically repeat the bandwidth prediction. Here, we repeat the probing and prediction with a period of 1 second.

In packet pair probing, the sender transmits two probe packets (of identical length), back to back, to a receiver. The receiver measures the delay between the probe packets and returns this information to the sender. This delay gives an indication of the current network load, a high delay indicates a high network load and vice versa. Formula 1 shows our simple calculation for the measured bandwidth.

$$BWT = L/\Delta_T \tag{1}$$

Where BWT is the resulting bandwidth, L is the probe packet length, and Δ_T is the measured delay between the probe packets $(T_2 - T_1)$.

In order to predict the future available bandwidth we use *exponential averaging*, which is a technique used to examine and average a sequence values along a time series which enables us to make a prediction based on previous predictions as well as the current network load.

Formula 2 shows how we predict the bandwidth:

$$P_k = \alpha BWT_k + (1 - \alpha)P_{k-1} \tag{2}$$

Where P_k is the future prediction, P_{k-1} is the previous prediction, BWT_k is the current bandwidth measurement, and α is a constant used to determine how important the history vs. the current measurement is in the prediction.

We implemented the bandwidth predictor as a user level program running on Linux running the 2.6 kernel. The program uses the *libpcap* [6] library to transmit and receive the probe packets directly to the network card (bypassing the higher layers). On the sender side, the program transmits the two probe packets back to back and then waits for a return packet containing the time difference measured at the receiver. The receiver waits for the two probe packets, takes time stamps when they arrive, and finally calculates the difference which is then transmitted back to the sender.

4 Traffic Shaping

The *Traffic Shaper* shapes the outbound traffic according to the portion of available bandwidth assigned to the node. It also prioritizes real-time over non real-time traffic for transmission.

Since we want to shape the outgoing traffic at a bit level we insert the traffic shaper as close to the network card as possible. At this position all outbound packets can be caught and queues before being processed by the traffic shaping mechanism.

The traffic shaper takes the predicted bandwidth as an input parameter and adjusts it's output rate accordingly. It also ensures that a node does not use more bandwidth than its allowed share. Figure 1 shows that

the traffic shaper architecture actually contains two *Token Buckets (TB)*, one for real-time and the other for non real-time traffic. Then we prioritize real-time packet transmissions by assigning a higher rate to the real-time TB .

We implemented the traffic shaper using the QoS features built into the Linux 2.6 kernel. The shaper architecture uses the *hierarchical token bucket* (HTB) and consists of a *u32* filter and two HTBs (real-time and non real-time) as shown in the traffic shaping part of figure 1.

The u32 filter allows us to distinguish between real-time and non real-time packets and send them to the corresponding HTB. When using HTBs a maximum transmission rate is set for the whole architecture (both HTBs). Transmissions rates are also set for both of the HTBs (with the sum equal to the maximum rate). The HTB implementation in the Linux kernel allows bandwidth unused by an HTB to be lent to another HTB, allowing for effective use of the bandwidth.

5 Status and Ongoing Work

We have implemented the architecture described above and currently we focus mainly on bandwidth prediction. First results from our implementation are encouraging.

We are currently investigating how to further enhance our bandwidth prediction by looking at more network parameters to be used as input [1].

Furthermore, we are investigating stability issues and the control aspect of the stream adaptation and the fluctuations of bandwidth on wireless networks.

References

[1] F. Carone and R. Guerra. Available Bandwidth Measurement on Wireless Networks. Technical report, Departement of Computer Engineering, 2004.

[2] A. Carpenzano, R.Carponetto, L. LoBello, and O. Mirabella. Fuzzy Traffic Smoothing: An Approach for Real-Time Comunication over Ethernet Networks. In *IEEE International Workshop on Factory Communication Systems*, Västerås, Sweden, August 2002.

[3] S. Keshav. A Control-Theoretic Approach to Flow Control. In *Conference of Communications Architecture and Protocols*, Zürich, Switzerland, September 1991.

[4] S.-K. Kweon, K. Shin, and G. Workman. Achieving Real-Time Communication over Ethernet with Adaptive Traffic Smoothing. In *IEEE Real-Time Technology and Applications Symposium*, Washington DC, USA, May-June 2000.

[5] L. Rizvanovic and G. Fohler. The MATRIX: A QoS Framework for Streaming in Heterogeneous Systems. In *RTMM - International Workshop on Real-Time for Multimedia*, Catania, Sicily, Italy, July 2004.

[6] tcpdump/libpcap, http://www.tcpdump.org/.

Influence of network awareness on perceived video quality.

Dmitri Jarnikov*, Peter van der Stok*‡, Johan Lukkien*

* *Dept. of Math. and Computer Science, Eindhoven University of Technology P.O. Box 513, 5600 MB Eindhoven, the Netherlands {d.s.jarnikov, j.j.lukkien}@tue.nl*

‡ *Philips Research Prof Holstlaan 4 5656 AA Eindhoven, the Netherlands peter.van.der.stok@philips.com*

Abstract

In this paper, we present a strategy to control a decoder that processes scalable video on a networked terminal. The controller maximizes user perceived quality taking into account available computing resources and condition of the network.

1. Introduction

We consider the problem of video streaming via a wireless medium. The medium is characterized by frequent and unpredictable quality fluctuations. Moreover, resource constrained terminals are not always capable of processing all video data that is transmitted by a sender. Scalable video can be used to overcome these problems.

Scalable video partitions video data into a Base Layer (BL) and one or more Enhancement Layers (EL). The transmission and decoding of the BL is enough to reconstruct video of recognizable quality, while the ELs are needed only for an incremental improvement of the quality of the received video sequence.

The variations in the number of received layers, resulting from instabilities of the network, along with resource limitations of the terminal require continuous decisions on the number of layers to be processed. A dynamic control mechanism for the networked terminal is proposed that uses a strategy created offline by means of a Markov Decision Process (MDP) [1]. The assumption of the model in [1] is that indifferent of the number of layers received for the current frame the probability to receive x layers for the next frame is $1/N*100\%$ (where N is maximal number of layers). Changing network conditions is not taken into account. Thus, the chosen strategy is equal for bad and good state of the media.

In this paper we present a method to create controller strategies for various network conditions. The network aware solution (Solution 2) is compared to the unaware one proposed in [1] (Solution 1) and it is shown that the network awareness of solution 2 results in fewer quality level changes in the system.

2. Model

A scalable video decoder [3] enables trade-off between resource consumption and video quality. Decoding BL and 1 to i ELs corresponds to quality level i. A change in a quality level results in a change of the video quality and resource consumption. Frequent changes of picture quality are not appreciated by user. The control strategy proposed in [1] minimizes the number of quality level changes while maximizing the average quality level.

The control strategy is a result of solving a MDP, which is constructed as follows. Decoding a frame has a deadline. The deadlines for the successive frames are strictly periodic. In each period the decoder is guaranteed a fixed amount of processing time, called *budget*. After decoding a frame, we calculate the relative progress, defined as the fraction of budget left until the deadline of the next frame. There is an upper bound on the relative progress that defines the maximal number of frames that can be decoded in advance. When decoding of a frame is finished, a decision must be taken about the quality level of the next frame. The maximal quality level is given by the number of layers received for the next frame as it is not possible to decode more layers than is available.

The decision of the controller depends on the number of layers available for the next frame and the amount of terminal resources, expressed in a value of the relative progress. Thus, we need to estimate how many layers will be available for the frame that will come after the next one. Solution 1 assumes that with equal probability it could be any number of layers. The proposed network-aware Solution 2 calculates separate probabilities for different combinations of number and size of layers as well as for various network conditions.

The probabilities of receiving a particular number of layers for the next frame are calculated by a network simulation based on the Gilbert-Elliot channel model [2]. The simulation takes into account layers configuration and network conditions. The network conditions are expressed in bit error rate (BER) and burstiness (μ).

3. Evaluation

We used a CE device with a hardware decoder to receive and decode video. Parsing and summation of the video layers are done in software. Processing BL only takes, on average, 4.4 ms. Processing BL and EL or BL and two ELs takes 12 and 24.8 ms respectively. As the influence of the content of a video on the hardware decoder is negligible, the time needed for processing a particular number of layers has a low deviation from the average.

We have chosen an 802.11b network as streaming medium, as it is the most widespread standard for in-home network nowadays. The network was used exclusively for our streaming application. There was no significant interference on the network, so the chosen parameters for the channel conditions were BER=10^{-6} and μ=0.3. The useful throughput of such a channel is around 6Mbps.

For the MDP we used the following parameters. The upper bound on relative progress is set to 2, which assumes that we can process one frame in advance. The utility function, which defines the reward for being on the particular quality level, is set to 2, 4 and 8 for quality levels 0,1 and 2 respectively. We set the deadline penalty to 100000. The quality change penalties for increasing the quality level are set to 5 and 50 if the quality level is increased by 1 or 2 respectively. For decreasing the quality level, the penalties are set to 50 and 500 for going down by 1 or 2 levels respectively. Solution 1 assigns no penalties for decreasing quality level if the decrease is caused by network conditions. For Solution 2 penalties for decreasing quality level are the same for changes caused by controller as well as by network.

For the comparison we've chosen 70000 frames long scalable video consisting of three layers: a BL and two ELs. We considered three different configurations with respect to the sizes of the three layers. In the first configuration all layers are 1Mbps, they consume only half of the available bandwidth, so ideally the controller strategy should be defined only by terminal resource limitations. The second configuration, with all layers of 2Mbps, requires all the bandwidth, so the network condition is as important as the terminal resources. The third configuration (all layers are 3Mbps) needs more bandwidth than is available, so the network condition should play the most important role in the controller decisions. Probabilities of receiving exactly one, two or three layers are given in Table 1. These probabilities are calculated offline by the network simulation.

Configuration	BL only	BL+EL_1	BL+EL_1+EL_2
1	0	0	100
2	0.08	51.36	48.56
3	71.29	28.71	0

Table 1 Probability (%) of having the given layers

We made pairwise comparison of the solutions, looking at the average quality level and the number of quality level changes. The comparison was made for all three above-mentions configurations. We considered budgets from 4 to 40 ms, with step of 1 ms.

For configuration 1, both solutions behave in the same way, delivering equal quality and making almost the same amount of quality changes. The reason is that in this configuration all three layers are constantly available for processing. Consequently, both solutions take into account only terminal resources (which are equal) resulting in nearly the same strategies.

The behavior of controllers for configuration 2 differs significantly (Figure 1,Figure 2). Starting from a budget of 12 ms, which allows successful decoding of two layers, the controller based on the strategy of Solution 2 does not attempt to increase the quality level. The reason is that according to the network conditions (see Table 1) the second EL is not available in, roughly, half of the cases. Thus, choosing quality level 2 will lead to frequent quality level changes. On the other side, the Solution 1 does not take network changes into account, resulting in higher average quality for the price of extremely high quality fluctuations.

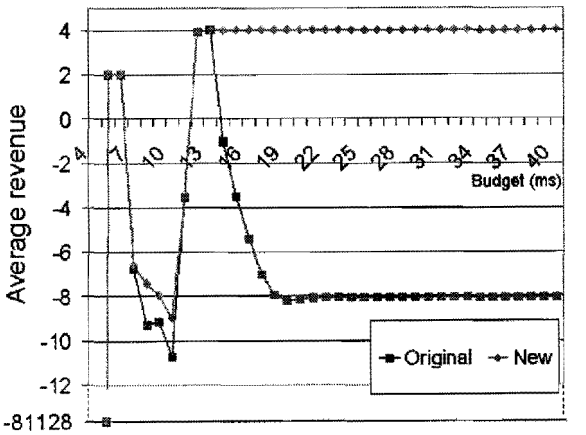


Figure 1 Average revenue for Configuration 2

The reason that the average revenue of both solutions is extremely low for the budget of 4 ms is a large number of deadline misses. The deadline misses occur because the budget given to the decoder is significantly lower than the average processing time for BL (4.4 ms). Starting with a budget of 5 ms, the number of deadline misses is 0, as there is always enough time to decode at least BL.

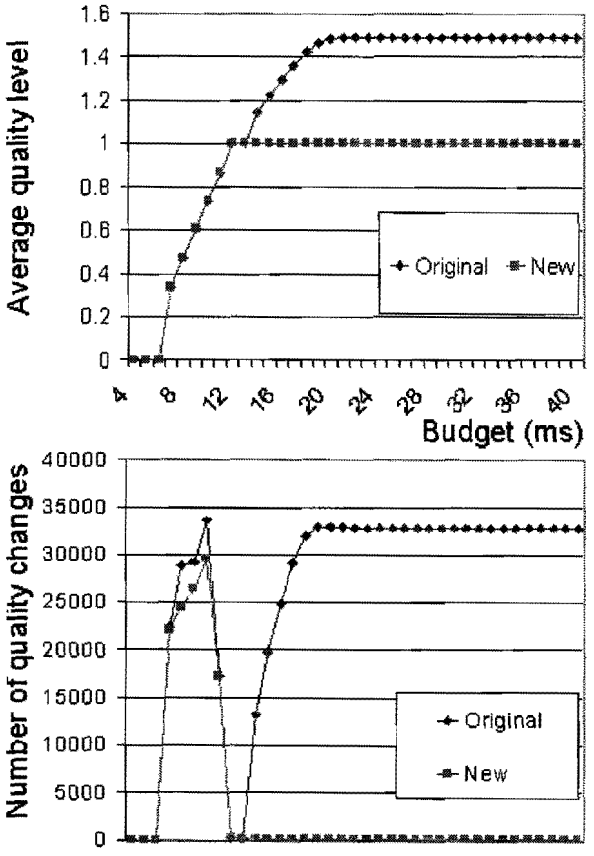


Figure 2 Comparison of average quality level and quality level changes for Configuration 2

The results for configuration 3 are again the same. Since receiving BL and two ELs is not probable, the controller is left with the choice between processing one or two layers. However, in view that most of the time (71.29%) only BL is available, both controllers behave conservatively, trying not to choose quality level 1. Thus, the strategy of the controllers is fully defined by the network conditions, as there are enough resources to decode BL for budgets higher than 4.4 ms (average decoding time for BL).

4. Conclusions

In this paper we presented improvements to the creation of the controlling strategy for scalable video decoder, presented in [1]. We've shown that the knowledge of the network conditions allows decrease of quality level changes, thus delivering a video with higher objective quality.

5. References

[1] Jarnikov, D.; van der Stok, P.; Wust, C.C., "Predictive Control of Video Quality under Fluctuating Bandwidth Conditions". Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on, Volume: 2 , pp. 1051 – 1054, June 27-30, 2004

[2] J. R. Yee and J. Edward J. Weldon, "Evaluation of the performance of error-correcting codes on a Gilbert channel", IEEE Trans. on Communications, pp. 2316-2323, Aug. 1995

[3] C. Hentschel, R.J. Bril, M. Gabrani, L. Steffens, K. van Zon, and S. van Loo, Scalable video algorithms and dynamic resource management for consumer terminals, In Proc. International Conference on Media Futures (ICMF), pp. 193–196, Florence, Italy, May 2001

Integrating cache space reservations with processing cycles reservations for a multiprocessor system on chip

Clara M. Otero Pérez, Jos van Eijndhoven
Philips Research Laboratories
{clara.otero.perez, jos.van.eijndhoven}@philips.com

Abstract

This article presents the initial steps towards integrating a cache space reservation mechanism with processing cycles reservations. Consumer electronics vendors increasingly deploy shared-memory multiprocessor SoCs, to balance flexibility (late changes, software download, reuse) and cost (silicon area, power consumption) requirements. The dynamic sharing of scarce resources by media applications jeopardizes robustness and predictability. Resource reservation is a well-known technique to improve robustness and predictability. Various resource reservations mechanisms address this problem individually for each resource such as processor cycles, cache space, and memory access cycles. However different resource types are very closely interrelated. We present a novel approach that aims to integrate the reservation mechanisms of the processing cycles and cache space.

1. Introduction

Multiprocessor systems on chip (SoC), such as Philips Nexperia [1], are rapidly entering the high-volume electronics market. Progressive IC technology steps reduce the impact of programmable hardware on the total silicon area and power budget. This permits SoC designers to shift more and more functionality from dedicated hardware accelerators to software, in order to increase flexibility and reduce hardware development cost.

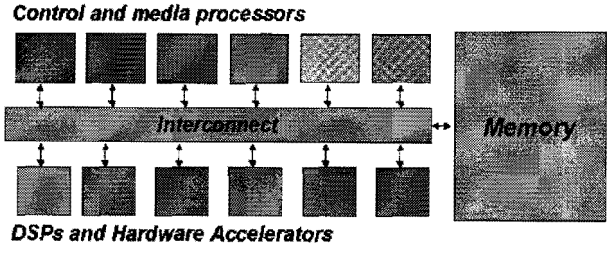


Figure 1 Heterogeneous SoC architecture with CPUs, DSPs, and accelerators communicating through shared memory.

However, these multiprocessor SoCs still combine flexibility—in the form of one or more programmable central processing units (CPU) and digital signal processors (DSP)—with the performance density of application-specific hardware accelerators. Figure 1 depicts such a heterogeneous SoC architecture as presented in [2].

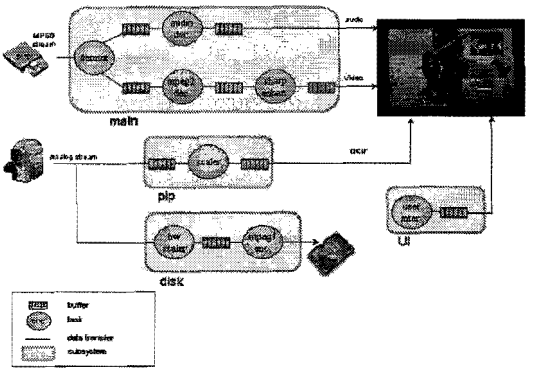


Figure 2 Media application example.

Figure 2 depicts an example of a streaming application [3] (mainly audio and video) that would run on such SoC. Our execution model for streaming applications consists of a connected graph in which the nodes represent either a task or a buffer. The interconnections represent the data transfer. The execution model is hierarchical. At higher levels of abstraction, a connected graph can again be viewed as a subsystem in a connected graph. Figure 2 depicts four such subsystems: main, pip, disk, and user interface (UI). The subsystems are denoted with the rounded rectangles.

2. Resource reservations

The sharing of resources is a potential source of interference that leads to unpredictability, and jeopardizes overall system robustness. Resource reservation is a well-known technique to improve robustness and predictability. It is based on four components, admission control, scheduling, accounting, and enforcement. When properly combined they provide guaranteed reservations.

Based on this approach, our system resource manager provides resource budgets to the subsystems. A *resource budget* is a guaranteed resource allowance. Guaranteed resource budgets provide temporal isolation, which contributes to robustness by bounding the interference caused by resource sharing.

The different resource types present in a SoC are very closely interrelated: for instance, a task running on a processor needs cache space to be able to execute. Therefore, the resource manager should address all shared resources in an integrated manner. The initial ideas towards this integration were presented in [4]. In this paper we address the integration of cache reservations with processing cycles reservations.

2. Cache space reservation

Figure 3 details the data path of a multiprocessor such as in Figure 1, in which a number of DSPs, CPUs, and accelerators communicate through shared memory, described in [5]. The architecture applies a two-level cache hierarchy to reduce memory bandwidth and latency requirements. The cache hierarchy is inclusive: a memory block can only be in a L1 cache if it also appears in the L2 cache. When a processing unit produces new data and stores it in its L1 cache, the L2 copy of that memory block becomes stale; in such cases a cache coherence protocol ensures that any consumer of the data always receives the updated L1 copy of the data (from L2). At any moment in time, a modified data item resides only in one of the L1 caches (and in L2). This property is intended to facilitate the partitioning of subsystems, consisting of multiple producer/consumer tasks, over multiple processors.

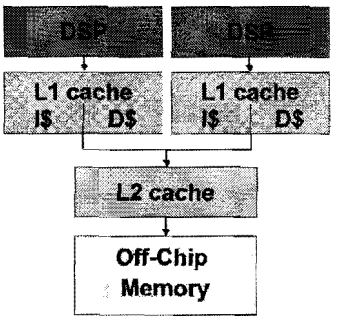


Figure 3 Data path for the memory hierarchy.

Traditional caches have been designed to work well for a single subsystem running on a single processing unit. In current systems, multiple subsystems execute concurrently sharing the L2 cache. These concurrent subsystems influence each other's performance by flushing each other's data out of the cache. Among the replacement and allocation techniques proposed, some of them use the concept of budgeting (or reservations). A given subsystem/task/thread has exclusive access to a specific part of the cache and will not suffer interference from other subsystems, which also have their own piece of cache. In [6] and [7] we can find examples of these budgeting which are spatial budgets.

Spatial budgeting improves subsystem performance by improving cache predictability. Furthermore, it enables compositionality of software subsystem. However, in resource constrained system the cache is also a scarce resource. This means that when a subsystem requests a cache budget, this cache space may not be available. In general, subsystems will not receive as much cache space as they require, with the derived performance penalty.

2. Processing cycles reservations

CPU-cycle budgets are provided to subsystems and must match the CPU cycle requirements of the subsystems. Media processing subsystems typically require periodic budgets with a budget value C (number of processing cycles), a granularity T (period of activation), and a deadline D. A periodic budget is replenished at regular intervals (T).

For lack of space but without loss of generality we explain in this section the simple case of periodic budgets and a reservation algorithm based on rate monotonic scheduling (RMS). This algorithm [8] was initially conceived for independently executing task. In our case, individual tasks are not independent whereas subsystems are. The same reasoning that used to apply to tasks applies now to subsystem budgets. We use an admission control algorithm that corresponds to the scheduling algorithm being used. If the admission control fails, the corresponding budgets cannot be guaranteed, therefore the subsystem corresponding to the budgets that causes the failure is not allowed to start (or to modify its resource requirements). When using RMS as scheduling algorithm a simple formula (1) for response time calculation from [9] is used:

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil * C_j \leq D_i \qquad (1)$$

In this formula, index i identifies the budget, T_i is the period, C_i is the budget capacity, R_i is the response time, and hp(i) is the set of all budgets with priority higher than i. The solution to this equation is the response time for task i. If the response time is smaller than the deadline the subsystem budget is guaranteed.

3. Integrated approach

To present the first step towards integrating the L2 cache resource reservation with processing cycles

reservations assume a system with three independent subsystems, such as Main, Disk and UI from Figure 2, executing on two processors. Subsystem one and two execute on processor one and subsystem three executes on processor two. Note that no data is shared among the subsystems. The budget scheduling algorithm is rate monotonic scheduling. Figure 4 depicts the resulting execution behavior and the corresponding cache reservation. In this case the cache reservation for each subsystem is 1/3 of the total cache.

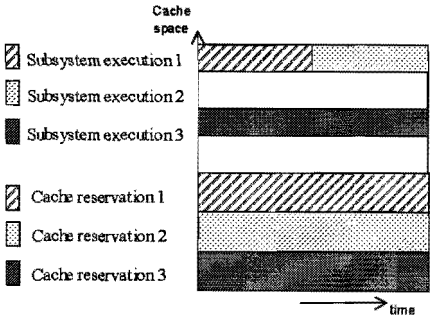


Figure 4 Cache reservation without using processing budget information

The cache reservation mechanism is controlled in software. We can use this mechanism dynamically by reserving cache space when the subsystem needs it and freeing (reallocating) it when the subsystem does not need it. The cache allocation decisions are made in software, and are enforced by novel extensions in the hardware cache controller. The difference with previous work is in the definition of "when the subsystem needs it". Until now the subsystem needed the cache space during its life time. In Figure 4, subsystem 1 keeps its cache reservation until the end of the period, even when it will not use it. Knowing that a subsystem will not execute for some time is not easy in the general case. However, if a processing budget is also provided then we can calculate exactly when a subsystem starts executing and when it will finish. The processing budget is available with granularity much smaller than the lifetime of the subsystem. For example a processing budget of 5 milliseconds each 10 milliseconds with respect to the lifetime of a subsystem of several hours.

Knowing the scheduling mechanism, and assuming equal periods for the processing budget we can calculate the worst case busy period per subsystem per processor. For the highest priority subsystem the start time is 0 and the finishing time is the response time calculated by (1). For the other subsystems, the start time is the response time of the adjacent higher priority subsystem and the finishing time is calculated by (1). In a more general case the earliest start time and latest end time can be calculated using the formulas from [10].

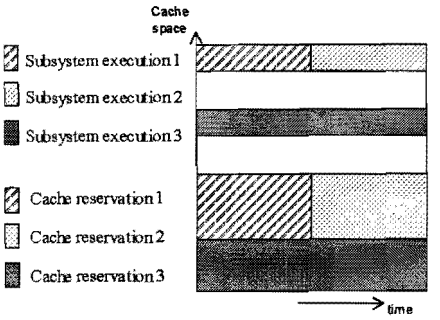


Figure 5 Optimized cache reservation by using processing budget information

Calculating the worst-case busy period, we can use the disjoint busy periods to maximize cache budget provision. In Figure 5 we can see how the cache space used by subsystem 1 is freed to be used by subsystem 2, maximizing overall cache space utilization.

This cache management mechanism is independent of (orthogonal to) other features like prefetching or invalidation for streaming data or multi-processor cache coherency.

4. Conclusion and future work

This article takes on the first challenge towards integrated resource management for SoC and presents a coherent approach that integrates cache space reservations with processing cycles reservations. By combining these mechanisms the cache space is more effectively used: when a subsystem does not need its reservation the space can be made available for other subsystems.

The current mechanism assumes a simple model where all budgets have the same period and fixed priority scheduling is used. We are working towards the extension of this mechanism towards the case of arbitrary periods and dynamic priorities.

4. Acknowledgment

The authors would like to thank Liesbeth Steffens, Martijn Rutten and Evert-Jan Pol for providing the initial ideas for this work.

6. References

[1] J.A. de Oliveira and H. van Antwerpen, "The Philips Nexperia™ Digital Video Platform," in *Winning the SoC Revolution*, G. Martin and H. Chang, Eds., pp. 67-96. Kluwer Academic, 2003.

[2] P. Stravers and J. Hoogerbrugge, "Homogeneous multiprocessing and the future of silicon design paradigms", *in Proceedings of the International Symposium on VLSI Technology, Systems, and Applications(VLSI-TSA)*, Apr.2001

[3] C.M. Otero Pérez, E. Steffens, G.v. Loo, P.v.d. Stok, R. Bril, A. Alonso, M. Garcia Valls, and J. Ruiz, "QoS-based resource management for ambient intelligence," in *Ambient Intelligence: Impact on Embedded System Design,* T. Basten, M. Geilen, and H. de Groot, Eds., pp. 159-182. Kluwer Academic Publishers, 2003.

[4] C.M. Otero Pérez, M. Rutten, E. Steffens, J. van Eijndhoven and P. Stravers, "Resource Reservations in Shared Memory Multiprocessor SoC," in *Dynamic and robust streaming between connected consumer electronic devices,* P. van der Stok, Ed., 2005.

[5] J. van Eijndhoven, J. Hoogerbrugge, J. Nageswaran, P. Stravers, and A. Terechko, "Cache-Coherent Heterogeneous Multiprocessing as Basis for Streaming Applications," in *Dynamic and robust streaming between connected consumer electronic devices,* P. van der Stok, Ed., 2005.

[6] A. Molnos, M.J.M. Heijligers, S.D. Cotofana, and J. van Eijndhoven, "Compositional memory systems for multimedia communicating tasks", in *Proceedings of Design Automation and Test in Europe (DATE)*, Mar. 2005, Munich, Germany.

[7] I. Ravi, "CQoS: a framework for enabling QoS in shared caches of CMP platforms", in *Proceedings of the 18th annual international conference on Supercomputing*, pp. 257-266, ACM Press, 2004.

[8] C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment", *in Journal of the ACM*, vol. 20, no. 1, pp. 46-61, 1973

[9] M. Joseph and P. Pandya, "Finding response times in a real-time system", *in British Computer Society Computer Journal*, vol. 29, no. 5, pp. 390-395, Oct.1986

[10] R. J. Bril, "Real-time scheduling for media processing using conditionally guaranteed budgets", *PhD. Thesis, Technical University of Eindhoven*, 2004.

Hierarchical QoS Concept for Multiprocessor System-on-chip

Milan Pastrnak[a,c], Peter Poplavko[b,c], Peter H. N. de With[a,c], Jef van Meerbergen[b,c]
LogicaCMG Nederland B.V.[a], Philips Research Labs Eindhoven[b]
Eindhoven University of Technology[c]
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
M.Pastrnak@tue.nl

I. Introduction

Resource-constrained systems require that Quality of Service(QoS) is an integral part of the system design. In this paper, we provide a new QoS concept to be used in a large Network-on-Chip(NoC) setup for complex multimedia applications. The trend in video consumer systems is to use a heterogenous flexible platform in which many tasks are executed in parallel. For cost efficiency reasons, the platform cost and therefore resources are bounded, so that quality control of applications is inevitable. QoS management for Systems-on-Chip (SoCs) has been extensively studied for e.g. MPEG-4 3D graphics, wavelet coding, etc. [1] [2]. The proposed QoS management approach computes the resoure utilization as an algebraic function of the quality settings, based on e.g. the number of graphical triangles to be processed. However, in our target application domain, the execution time also depends on the input data. The input data dependency is particularly important for advanced multimedia applications, where video content is partitioned into independent video objects instead of rectangular frames of samples. As objects can change over time, in size, shape and texture contents, the processing requirements are more variable then with conventional video. For this reason we have adopted the MPEG-4 standard as an example of object oriented video processing.

In this paper we consider a class of QoS systems that relies on predicting the execution times of the application at run-time taking the data dependency into account. We propose a control mechanism based on a linear prediction model that interacts with a higher level QoS management unit to adopt quality and resource settings for our application. In initial experiments with our new concept, we found that the savings in computation is at least in the order of several tenths of percents.

[1]Supported by the European Union via the Marie Curie Fellowship program under the project number HPMI-CT-2001-00150.

II. Problem Statement

In this paper we study a complex multimedia application that can have very dynamic execution time characteristics per frame. In our previous work we motivated the use of a multiprocessor system-on-chip (MP-SoC) as a target platform for such applications [3]. MP-SoC is a reconfigurable platform and by using multiple processors executing a plurality of tasks, the QoS management becomes a multi-dimensional problem. In this paper we concentrate on the problem of mapping MPEG-4 shape-texture decoding as an example of a complex multimedia application. The object-oriented video is a more dynamic application in terms of resource usage (computation, memory, communication) than with regular video signals. The above-mentioned dynamism of processing of video objects in combination with the multiprocessor target platform, forms a complex mapping and run-time control problem. We concentrate on mapping MPEG-4 decoding on a distributed set of processors with the aim to control the complete application with respect to execution flow and output quality levels. Preferably, the mapping is scalable in terms of QoS and introduces a model for execution control.

III. QoS Model for Applications on MP-NoC

We focus on a complex state-of-the-art multimedia application (MPEG-4 shape-texture decoding) that can have very dynamic execution time characteristics per frame, and of which several instantiations can be executed in parallel. In our previous work we motivated the use of a multiprocessor system-on-chip (MP-SoC) as a target platform for such advanced applications [3], which makes also QoS management a multi-dimensional problem.

The architecture of our QoS concept is based on two negotiating managers, instead of the conventional single resource manager. We distinguish the *Global QoS manager* that controls the total system performance involving all applications and the *Local QoS manager* that controls an individual application within the assigned resources (see Fig. 1). Since the responsibilities of both managers are essentially different, it becomes apparent that a protocol for negotiation beetwen these two managers will be needed.

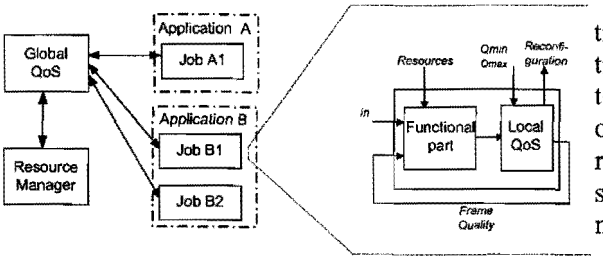


Fig. 1. Architecture of the proposed QoS.

Each application is divided in jobs and the platform should support each job for implementation, using the system resources and the software libraries. A job is split into different tasks, e.g. the MPEG-4 header parsing, the inverse quantization, IDCT, etc. In order to execute a job on a multiprocessor, tasks are mapped onto processors and other resources (memory, communication resources, etc.).

The use of the above architecture for system quality control makes sense if an accurate execution model for the application mapping can be provided. Our previous studies resulted in a quite accurate prediction model based on a linear parametrical equation at task level [3]. Nevertheless, not all of these parameters are *a priori* available when the task starts its execution. In [4], we proposed a prediction technique that estimates resource usage based on a set of parameters available from video headers and a complementary set of parameters from the decoding of previous frames. The resulting task level model offers continuous control of the decoding process with an acceptable accuracy, but at a finer granularity than current frame-based QoS systems.

Our mapping strategy exploits the predictability property of our architecture to enable a deterministic Quality-of-Service (QoS) from each job, independent of other jobs. To achieve this, we reserve resources for each particular job in the form of virtual processors and virtual connections. These virtual processors and connections are run-time assigned to the existing resources of the platform. This abstraction is important to obtain the worst-case model of resource distribution in case when each task is mapped to a different processor of the platform, and to keep independency between jobs.

Advanced QoS control requires an accurate estimation of the resource usage. Therefore, we distinguish an *off-line phase* where jobs are mapped to virtual processors to obtain specific application operating points and *run-time refinement* of the resource usage based on the current resource-usage status of the system. Both phases are described in more detail below.

A. Off-line: intra-job mapping

The purpose of the intra-job mapping is to generate a set of operating points, which allows to online trade-off between the quality resulting from the job and the resource usage by selecting an appropriate operating point. For each operating point, a certain quality setting is initially assigned. Afterwards, a set of virtual processors and connections are allocated. Different tasks are inserted in sequential order into allocated processes, and the processes are partitioned over the virtual processors. The data transfers between the virtual processors are assigned to the virtual connections. The result of allocation and assignment is a *virtual platform* for the job, and a network of concurrent communicating processes

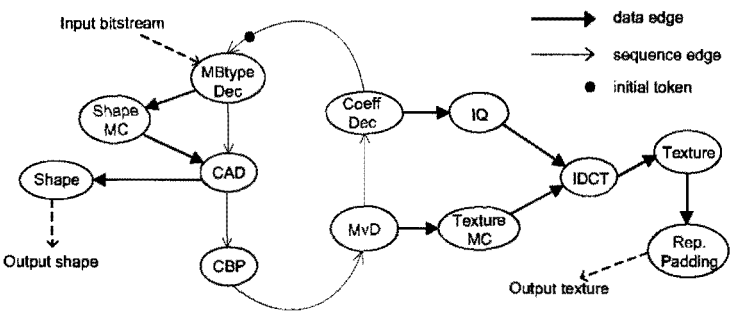


Fig. 2. Computation graph of distributed MPEG-4 arbitrary-shaped video-objects decoder.

for the job that is mapped onto the virtual platform. This network is called a *configuration network*. A major objective of intra-job mapping is to create a virtual platform using minimum resources. On the other hand the platform should offer enough resources such that the deadline miss rate of the job is low enough. Each operating point is defined by a quality setting and a virtual platform with the corresponding configuration network.

The quality setting gives only an estimate of the average optimal quality setting for the given mapping. Due to variation of the execution time, at run-time the quality settings should be adjusted continuously. The corresponding control system is called Local Quality Manager and is discussed in the Section IV.

B. Online: Resource management

The resource manager controls the available physical resources in conjunction with the Global quality manager, thereby using the operating points generated off-line. This works as follows. For a starting job, the Global quality manager chooses an initial quality setting by selecting an operating point. Based on off-line measurements of the anticipated quality Q, the manager strives for a quality setting that would satisfy the user. At this point, the resource manager is of key importance, as it keeps track of the free capacity of all physical resources in the platform. Given a virtual platform, for each virtual processor the manager should find a physical processor with sufficient free capacity. For each virtual connection, free network resources should be found. It may happen that the resource manager cannot accommodate the resources for the new job. If the job has a high importance, the Global quality manager may decide to decrease the quality settings of some other jobs to release resources for the new job.

IV. QoS for Arbitrary Shaped Video Object Decoding

We have studied object-oriented MPEG-4 coding for the new proposed QoS concept. In MPEG-4, every Video Object (VO) is represented in several information layers, with the Video Object Plane (VOP) at the base layer. This VOP is a rectangular frame containing hierarchically lower units, called Macroblocks (MB). Figure 2 outlines a distributed version of a computation model for arbitrary shaped video object decoding at the finest granularity (MB level). In this graph, each task starts its execution when it has data on all inputs (as in Homogeneous Dataflow Graphs). Our application model also offers options to skip some of the tasks under certain conditions (e.g. repetitive padding is sometimes not needed).

Figure 1 portrays the Local QoS connected with a functional part of the actual Job. A runtime steering mechanism periodically observes the difference between the estimated resource requirements and the actually used resources. Based on the estimation error and input data characteristics, Local QoS handles two types of situations. First, short-term variations of the resource utilization may occur. To compensate for it, in the case that some of the tasks are scalable, the Local QoS manager

can change their local quality settings. If this step does not reduce resource needs sufficiently, it can disable some steps of the decoding process (e.g. a deblocking filter). Second, when the change in required resources has a long-term nature that needs control, e.g. the size of an object changes significantly, Local QoS can request for reconfiguration and new reservation of resources to the Global QoS management unit.

To evaluate the performance of QoS management, performance is modeled into an objective function based on quality settings and deadline miss rate. If the Local QoS estimates a deadline miss for presentation, but the next frame depending on the decoded frame will meet the deadline, the decision unit of the Local-QoS manager executes the decoding and omits the presentation. The objective function for performance estimation is

$$f = 1-(M_r+S_r)-N_r+0.5N_{r1}+0.25N_{r2}..., \quad (1)$$

where M_r stands for the rate of frames that were decoded but never presented, S_r is the rate of frames skipped completely and N_r represents the rate of frames that were presented but not on time. We can extend our function with a more detailed focus on frames displayed two frames after expected (denoted as N_{r1}), or three frames (N_{r2}), etc. Another possible extension is to add the weighted effect of changing the quality of scalable tasks internally of the job. However, for the evaluation of our QoS concept, Equation (1) provides sufficient means.

V. Conclusions and Future Work

We have proposed a new QoS management system supporting the control part of a resource-constrained system. The QoS system is split in two layers: Global QoS for overall system control and Local QoS for single application control and resource management. Our control systems are based on input-dependent models for video execution. The models exploit a parametrical linear timing specification of task-based subapplications. This approach was implemented in an experimental MPEG-4 decoder for arbitrary-shaped video objects. We have observed that our proposed QoS technique saves considerable amount of unused resources (up to 64%) compared to conventional QoS based on frame skipping when a deadline is missed. It should be noticed that the amount of saved resources depends on the contents of the test sequence and the utilization of the system. We found that prediction of resource utilization can lead to significant quality improvement of the complete system when resources are offered for other applications. The proposed QoS mechanism runs fast enough to be executed in real time. In future, we intend to combine data-dependent application models with event-based applications.

References

[1] J. Bormans, N.P. Ngoc, G. Deconinck, and G. Lafruit, *"Terminal QoS: advanced resource management for cost-effective multimedia appliances in dynamic contexts" in Ambient intelligence: impact on embedded system design*, pp. 183–201, Kluwer Academic Publisher, The Netherlands, 2003.

[2] R.J. Bril, *Real-time scheduling for media processing using conditionally guaranteed budgets*, PhD Thesis, Technische Universiteit Eindhoven, The Netherlands, September 2004.

[3] M. Pastrnak, P. Poplavko, P.H.N. de With, and D. Farin, "Data-flow timing models of dynamic multimedia applications for multiprocessor systems," in *4th IEEE Int. Workshop System-on-Chip for Real-Time Applications (SoCRT)*, 2004.

[4] M. Pastrnak and P.H.N. de With, "Data storage exploration and bandwidth analysis for distributed mpeg-4 decoding," in *8th IEEE Int. Symp. Consumer Electronics (ISCE)*, 2004.

TU/e technische universiteit eindhoven
Bibliotheek
Postbus 90159
5600 RM Einhoven
Tel: 040 - 247 22 24
200510730

WORKSHOP ON RESOURCE MANAGEMENT FOR MEDIA PROCESSING IN NETWORKED EMBEDDED SYSTEMS

ISBN 90-386-0544-7