

Soft Real Time Design

by *Gerrit Muller* Embedded Systems Institute
e-mail: `gerrit.muller@embeddedsystems.nl`
`www.gaudisite.nl`

Abstract

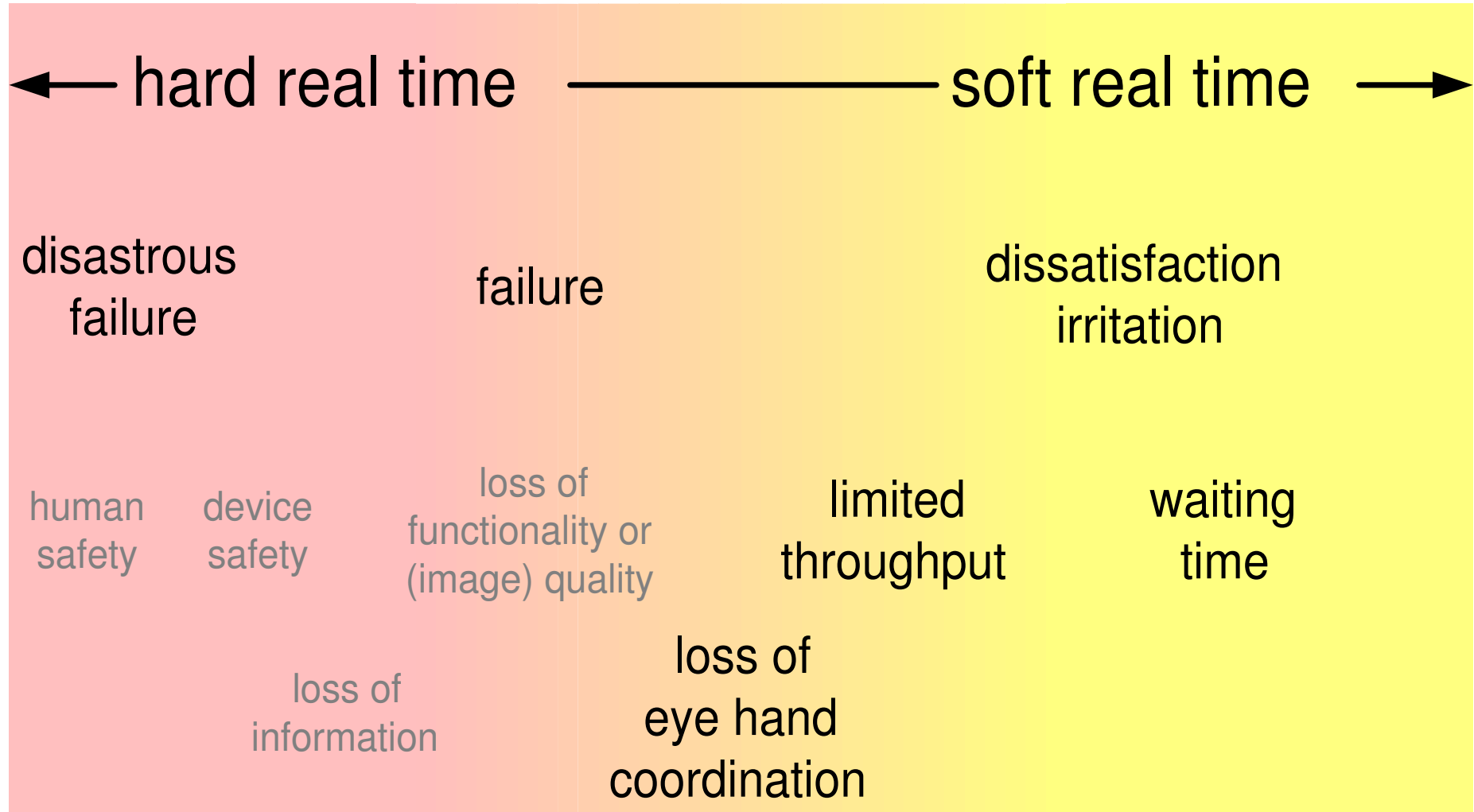
Soft Real Time design addresses the performance aspects of the system design, under the assumption that the hard real time design is already well-covered. Core decisions in soft real time design are:

- granularity
- synchronization
- prioritization
- allocation
- resource management

The complete course ASPTM is owned by Embedded Systems Institute. To teach this course a license from Embedded Systems Institute is required. This material is preliminary course material. The final material and course information can be found at: www.esi.nl/cursus.

February 10, 2011
status: preliminary
draft
version: 0.2

Soft Real Time Design



TV zapping

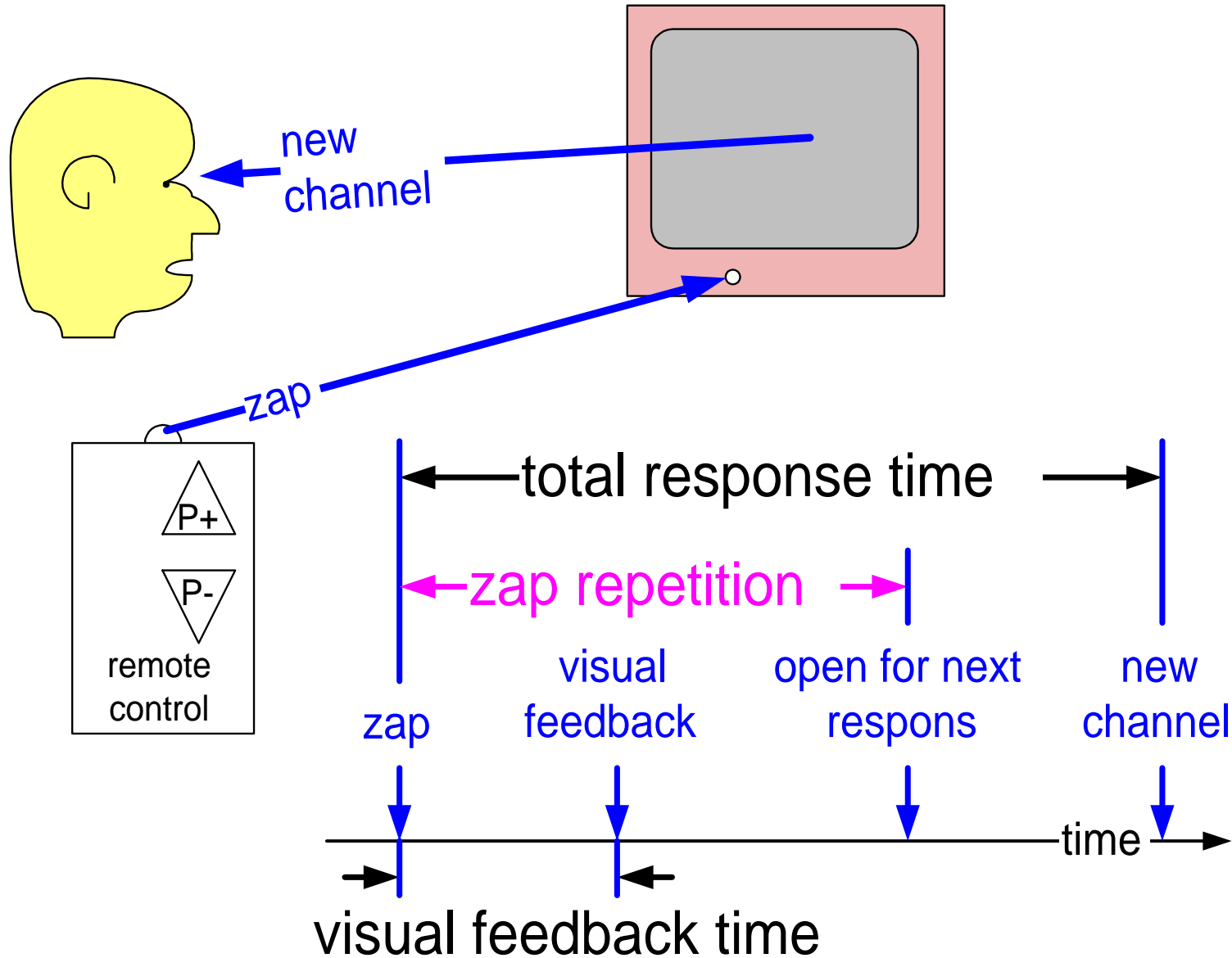
Problem introduction

Approach for solving response time problems

Revised functional model

Measuring and modelling

Zap timing: What is the Requirement?



Approach

1) Measure the end-to-end time

2) Decompose the processes

based on expected outcome

3) Measure the individual components

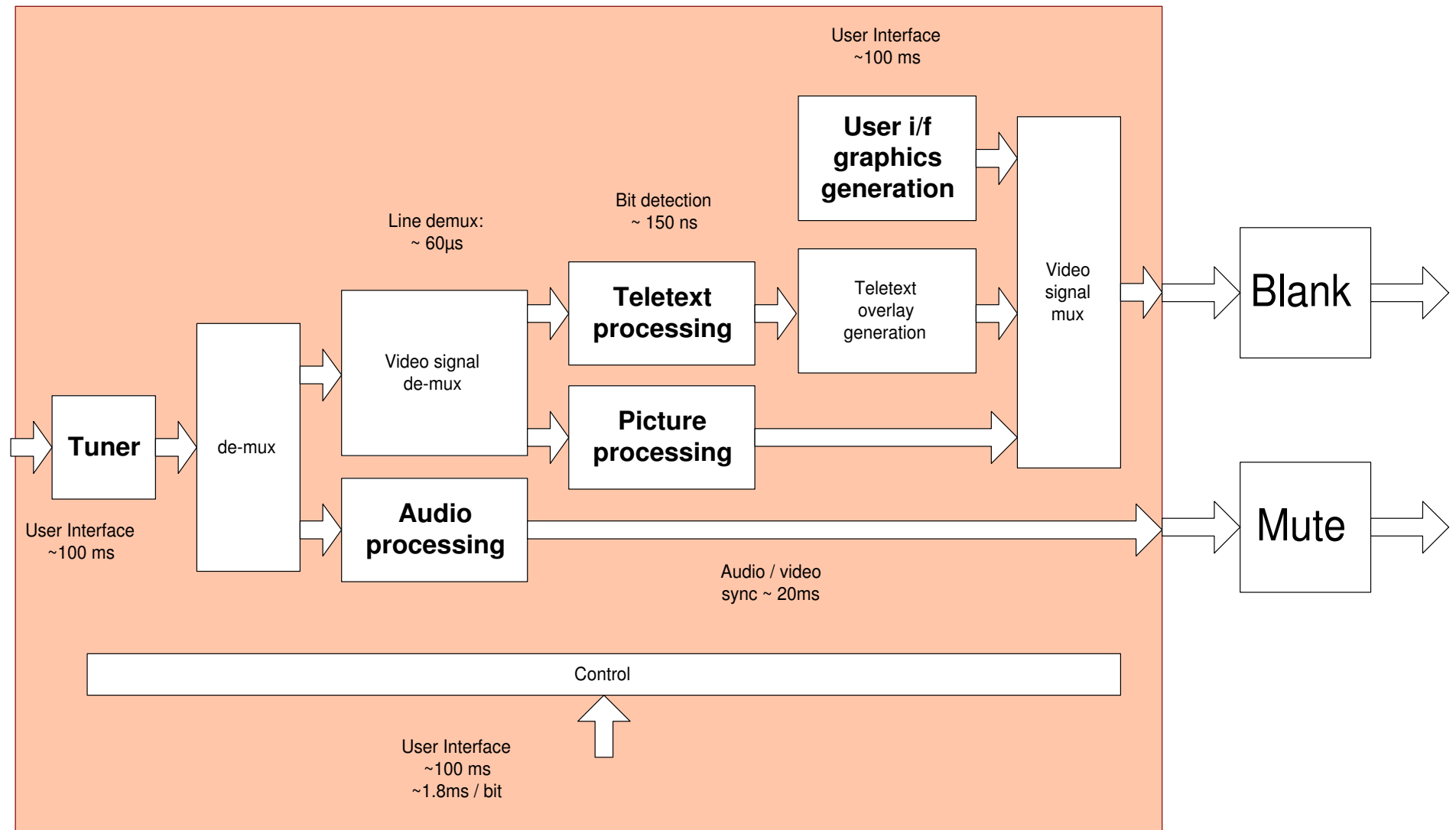
use previous decomposition (2)

4) Clarify the unknown parts and make them explicit

5) Further divide the major posts

6) Aggregate the smaller posts

Functional Model



Expected and Measured Values

Expected values:

Mute	: 50 ms
Blank	: 40 ms <i>1 frame</i>
Flush AV pipeline	: 160 ms <i>4 frames</i>
Set tuner	: 200 ms
Fill AV pipeline	: 160 ms <i>4 frames</i>
Unmute	: 50 ms
Unblank	: 40 ms

Measured values:

Mute	: 60 ms
Blank	: 120 ms
Flush AV pipeline	: 0 ms
Set tuner	: 180 ms
Fill AV pipeline	: 40 ms <i>1 frame</i>
Format detection	: 200 ms <i>5 frames</i>
Unmute	: 60 ms
Unblank	: 120 ms
Summing	: ~ 900 ms
Total time measured:	2000 ms

Zapping Problem step 4

Somewhere 1000 ms are missing

Detection of frame size takes a long time!

+ Lots of software overhead

Analyze frame size detection and SW overhead

Zapping Problem step 5

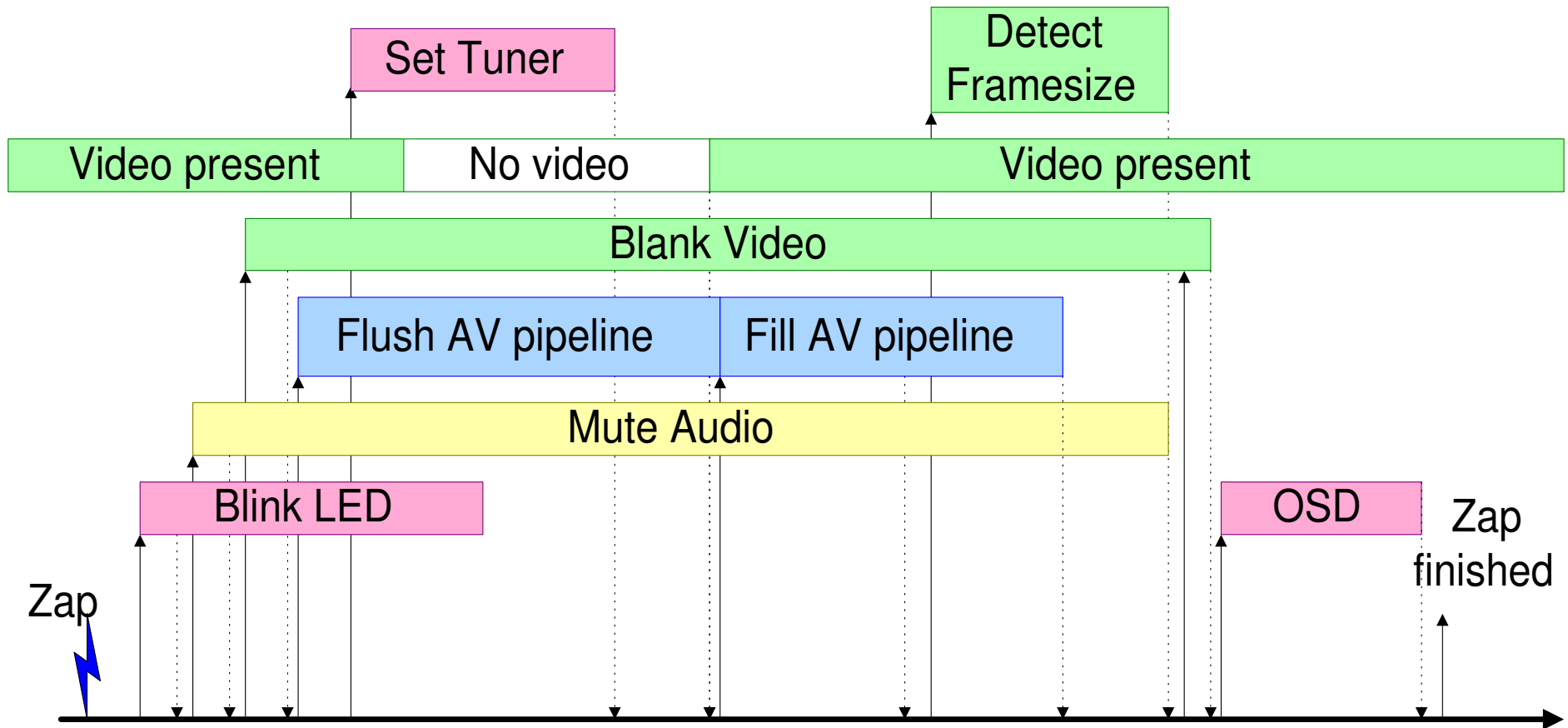
Subdivide / analyze format detection (200 ms)

Zapping Problem step 6

Ignore pipeline effects

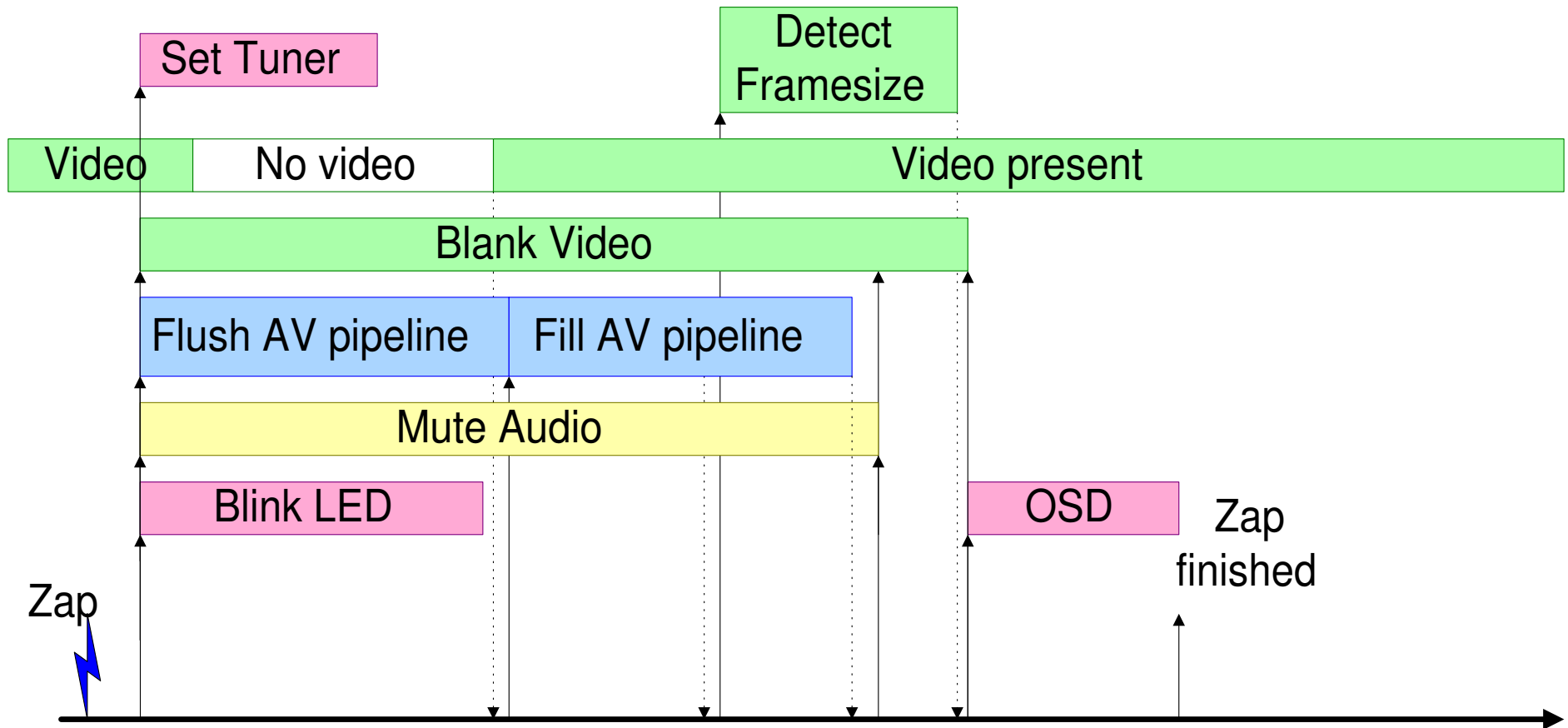
Simple Concurrency Model (with waits)

Zapping tasks sequential



Simple Concurrency Model (optimized)

Zapping tasks parallel



TV zapping

Understanding of the problem is crucial

Iterate over modelling and measuring to build balanced performance model

EasyVision: Resource Management

Introduction to application

SW design

Memory and performance

Memory design

CPU load and Performance

Easyvision

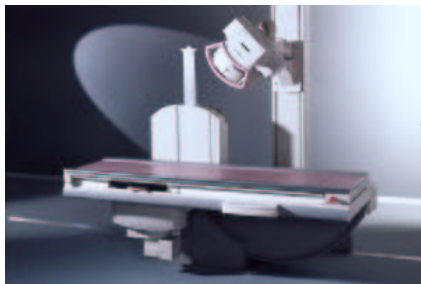
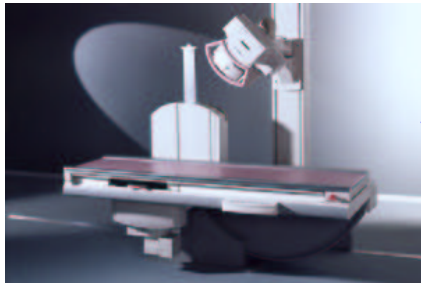
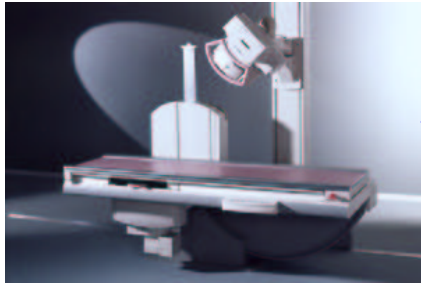
Medical Imaging Workstation

serving 3 X-ray examination rooms

providing interactive viewing and printing on high resolution film

Challenge: interoperability and WYSIWYG over different products

Easyvision Serving Three URF Examination Rooms



URF-systems

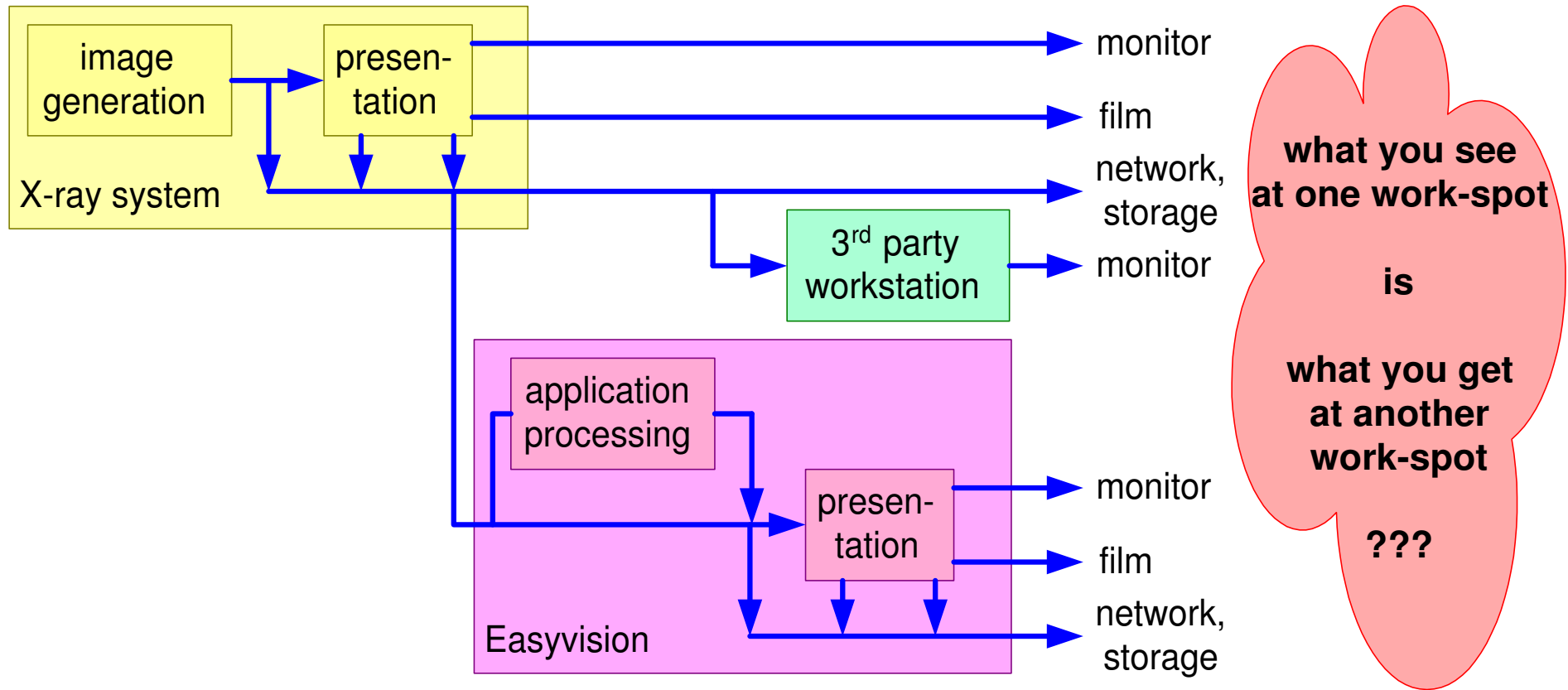


EasyVision: Medical Imaging Workstation

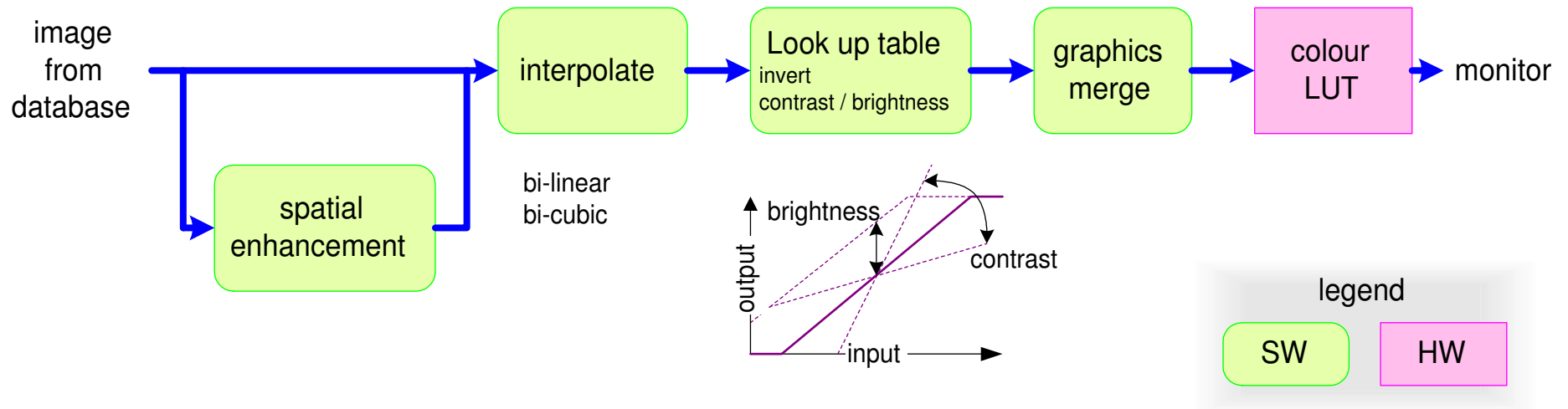


typical clinical image (intestines)

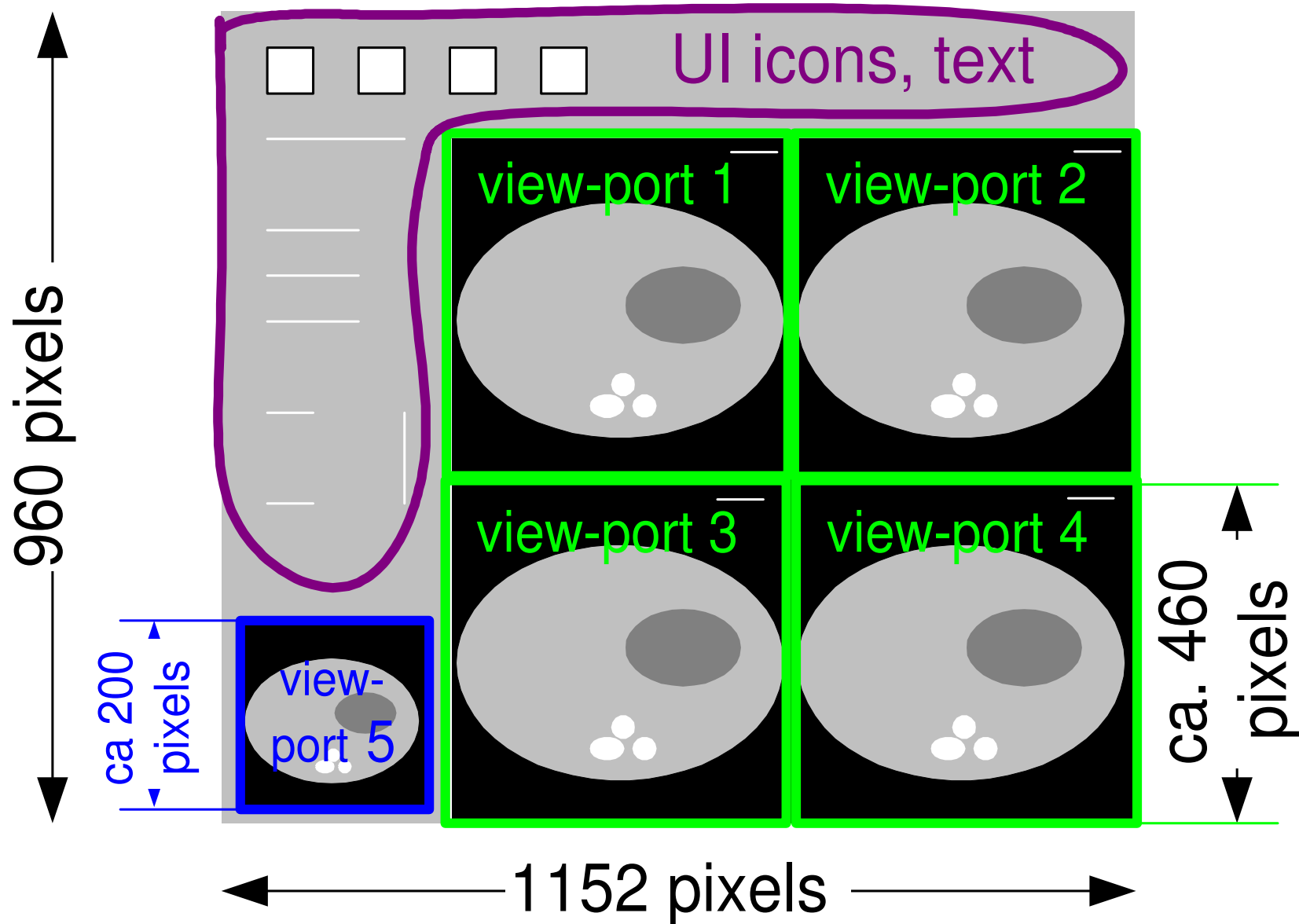
Image Quality Expectation WYSIWYG



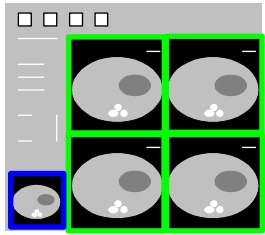
Presentation Pipeline for X-ray Images



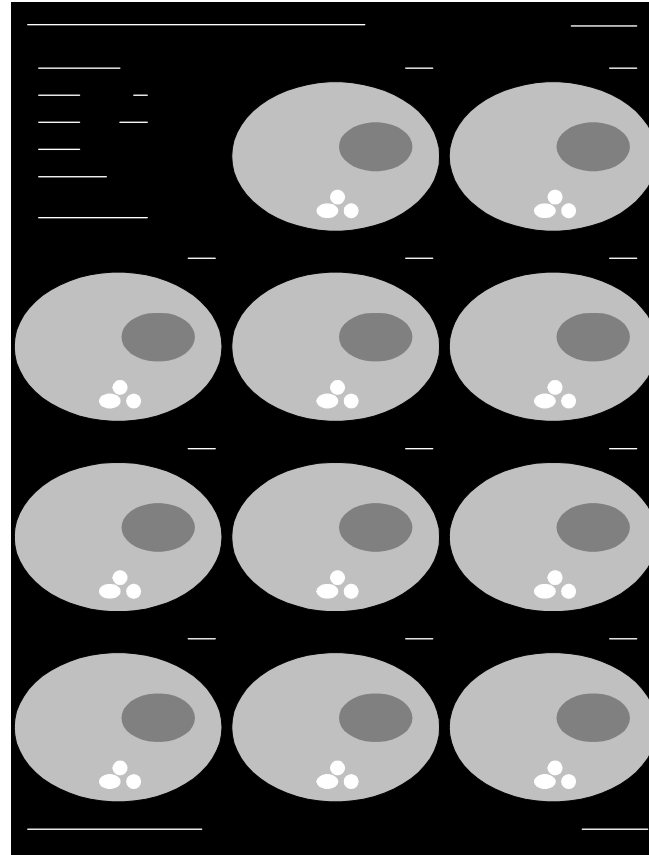
Quadruple View-port Screen Layout



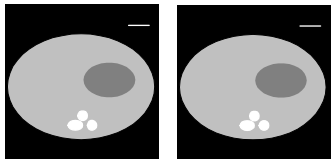
Rendered Images at Different Destinations



Screen:
low resolution
fast response



Film:
high resolution
high throughput



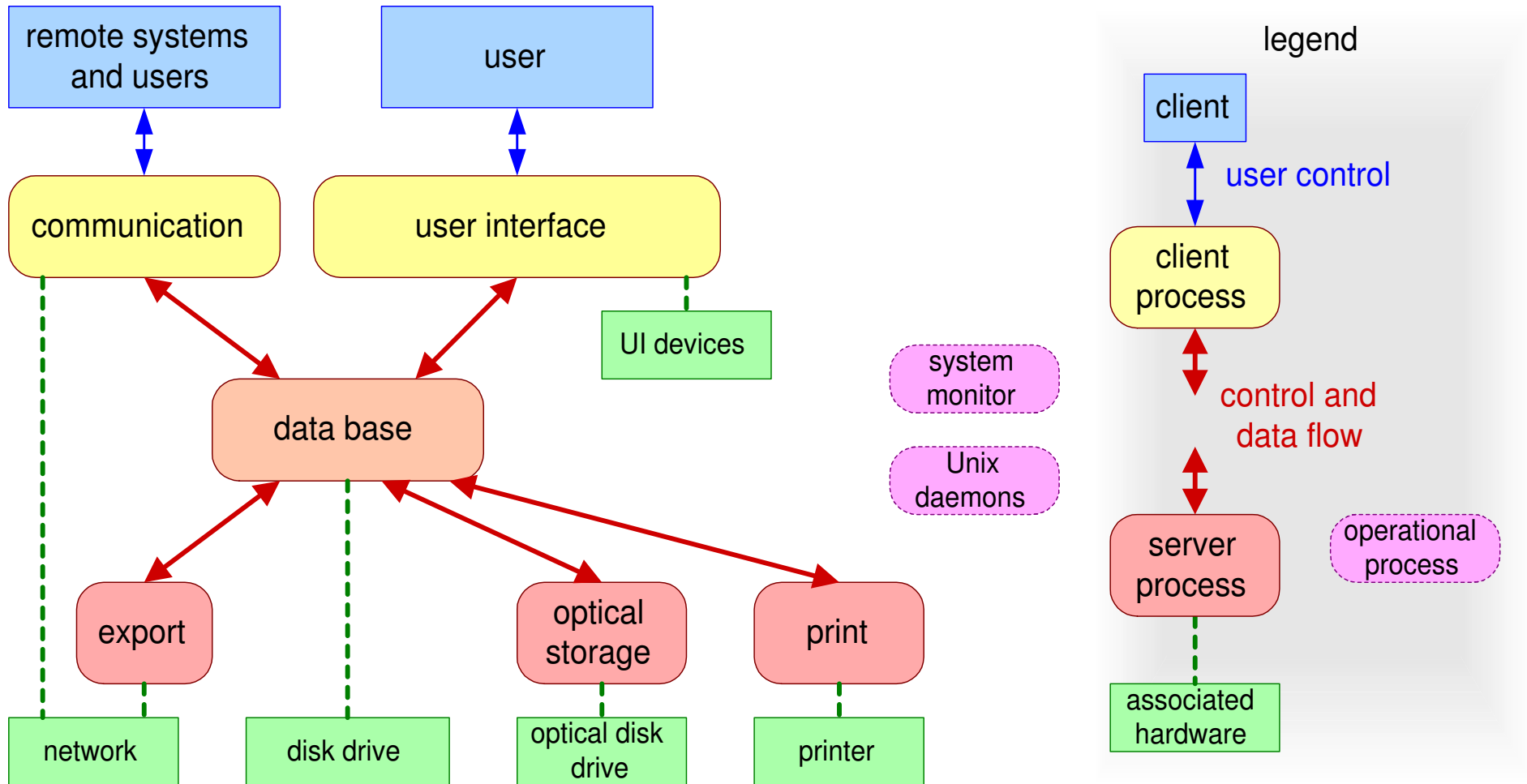
Network:
medium resolution
high throughput

Easyvision SW design

Concurrency design

SW layers

Concurrency via Software Processes

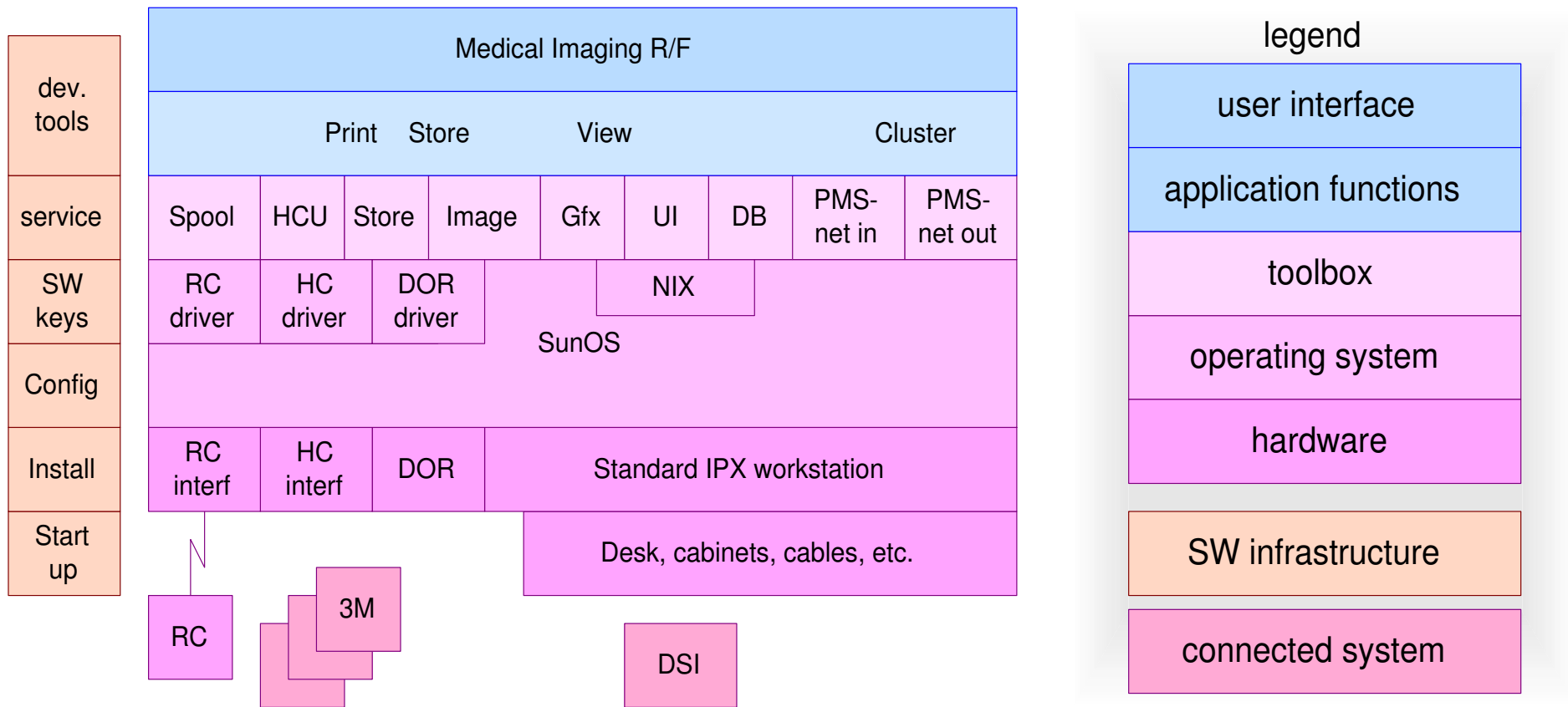


Criteria for Process Decomposition

- management of concurrency
- management of shared devices
- unit of memory budget (easy measurement)
- enables distribution over multiple processors
- unit of exception handling: fault containment and watchdog monitor

Processes are a facility provided by the Operating System (OS) to manage concurrency, resources and exceptions

Simplified Layering of the SW (Construction Decomposition)



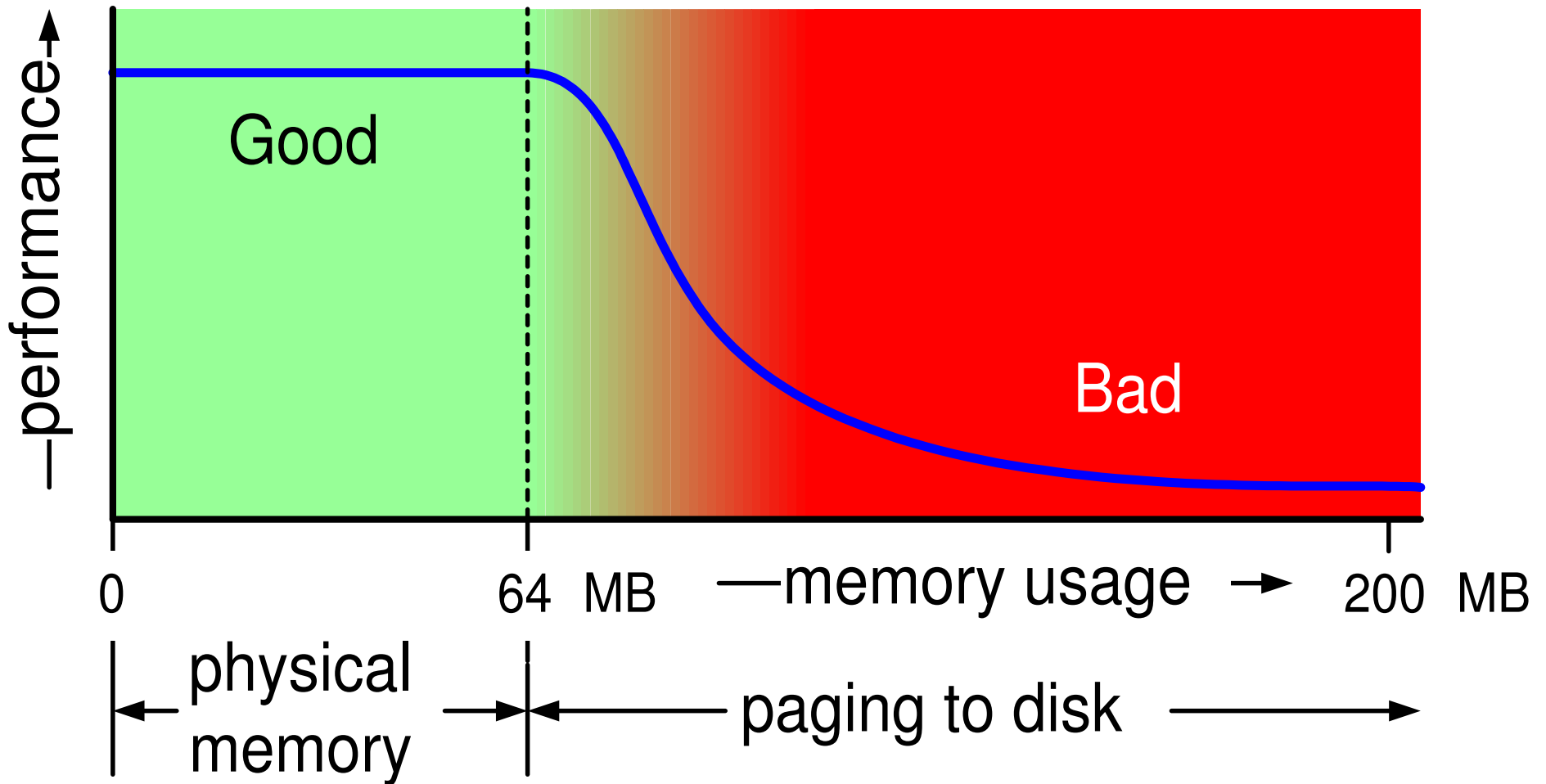
Easyvision Memory and Performance

Performance problems

Analysis of memory use

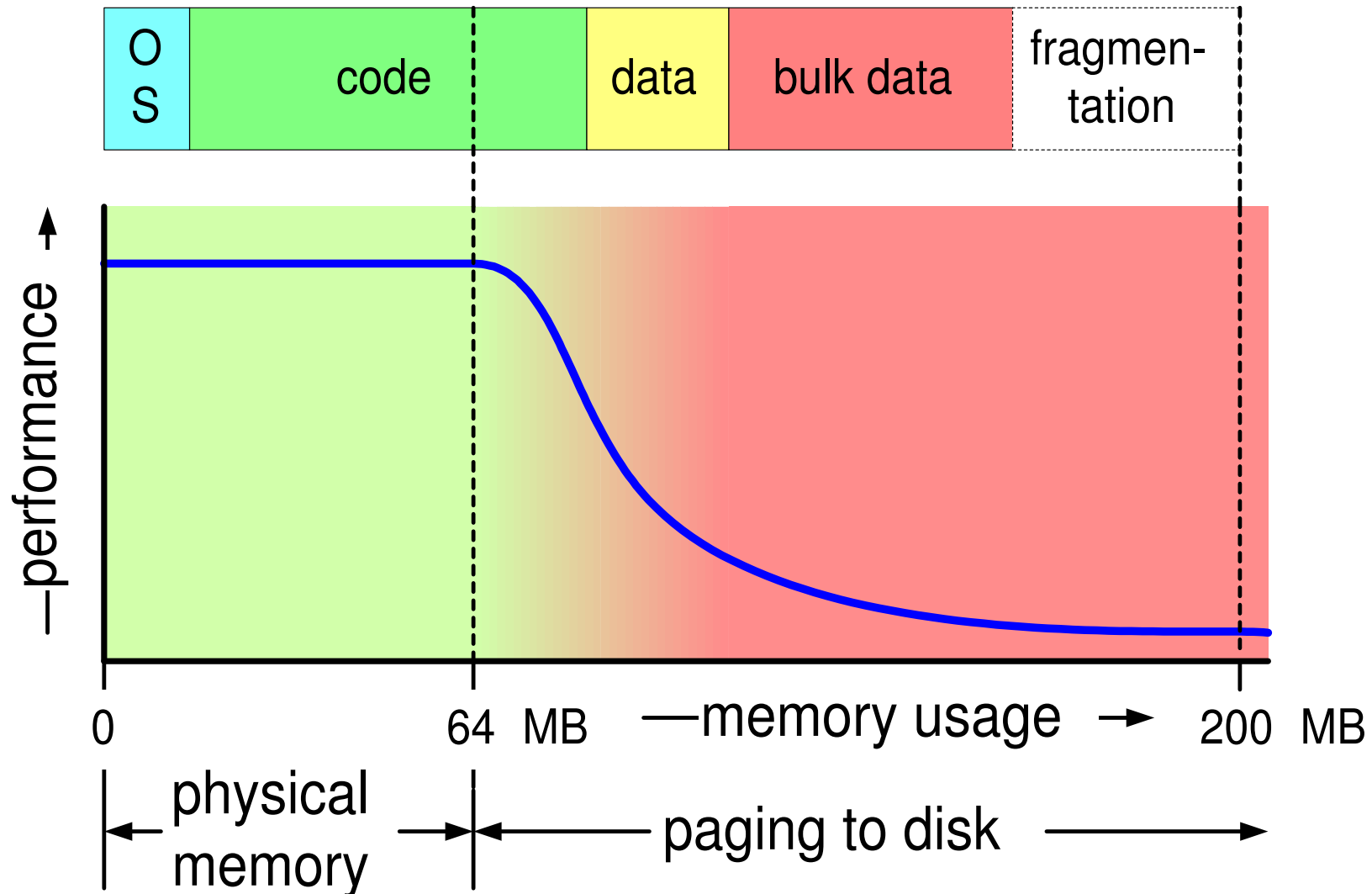
Memory budget

Performance as a Function of Memory Use

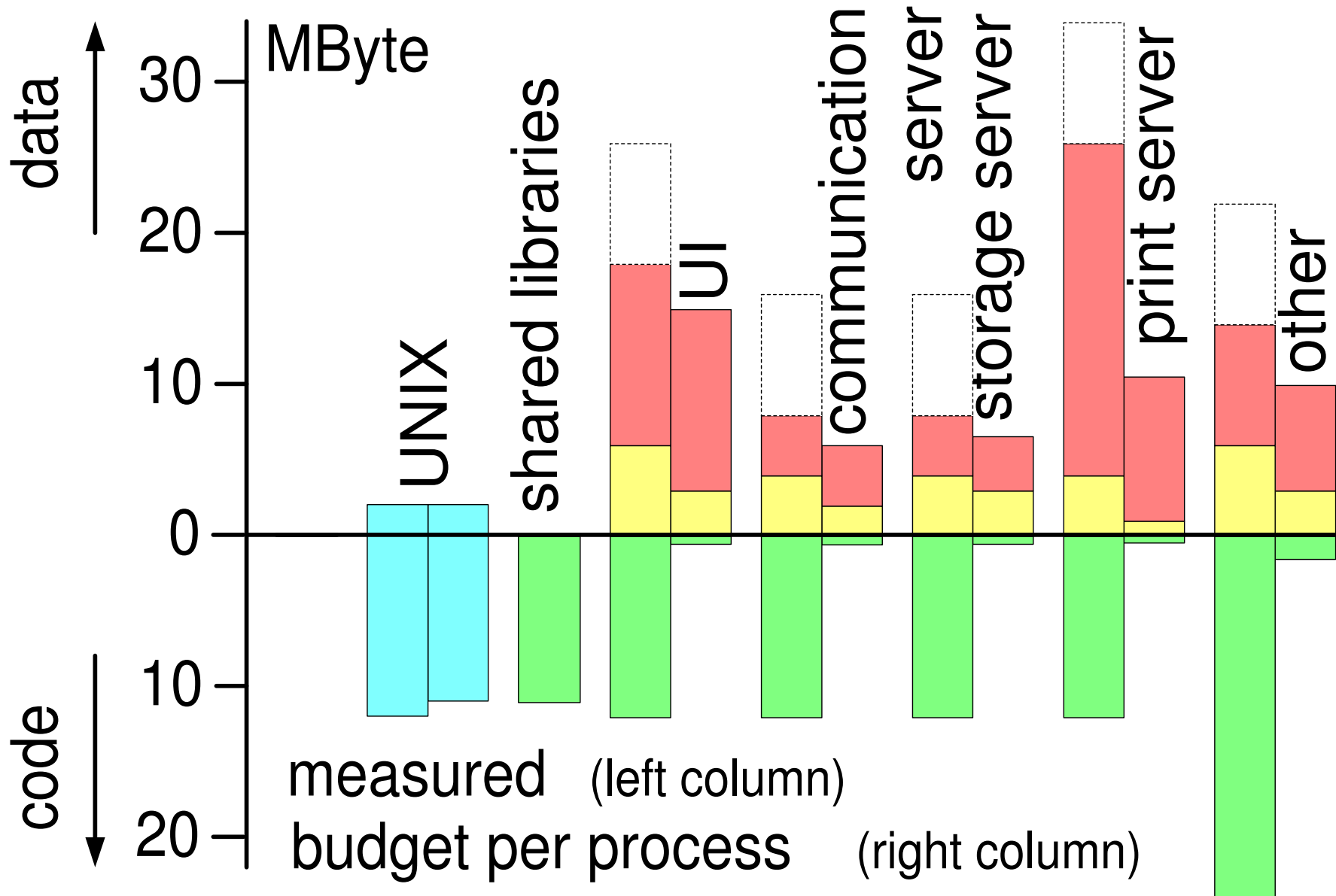


Problem: Unlimited Memory Consumption (1992)

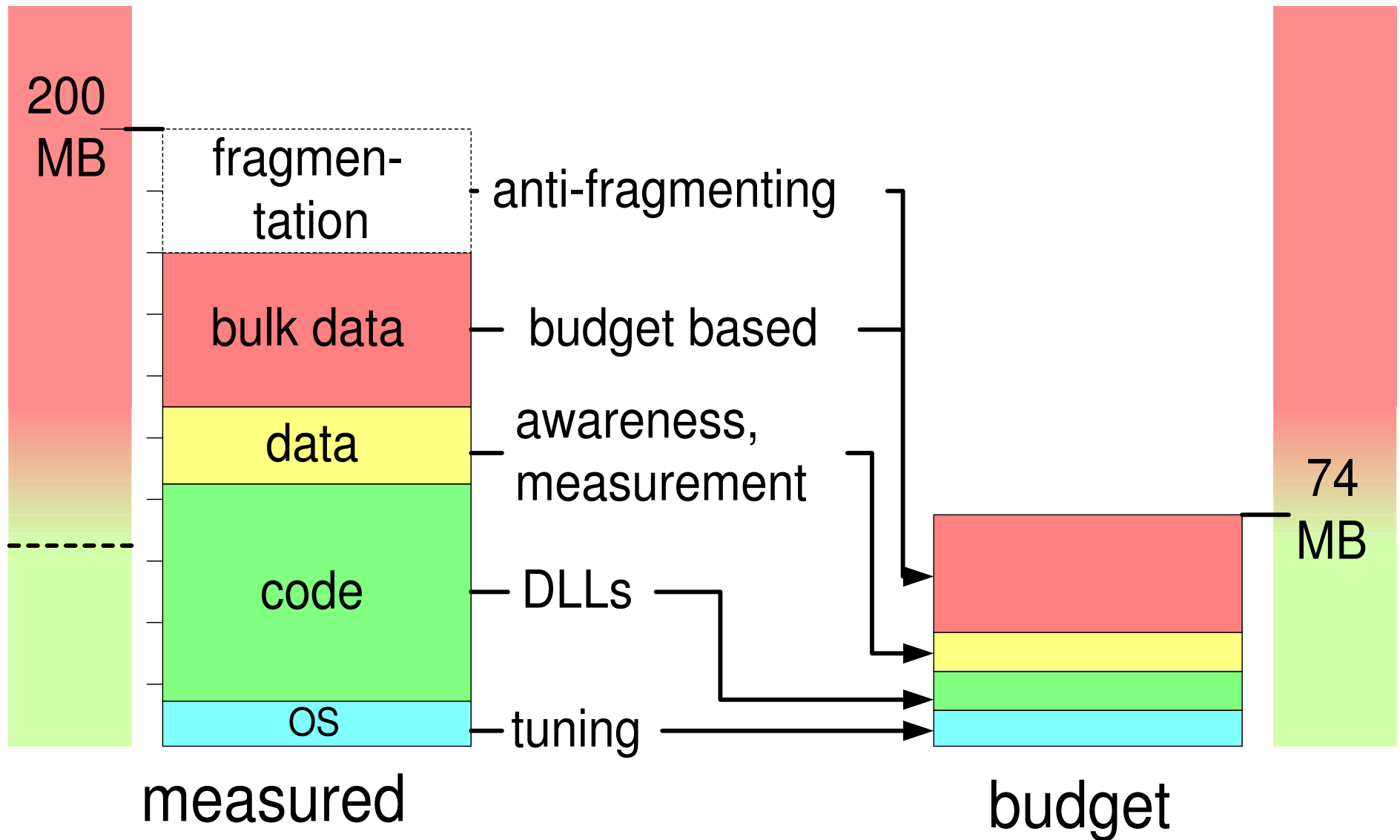
total measured memory usage



Measurement Per Process



Solution: Measure and Iterative Redesign



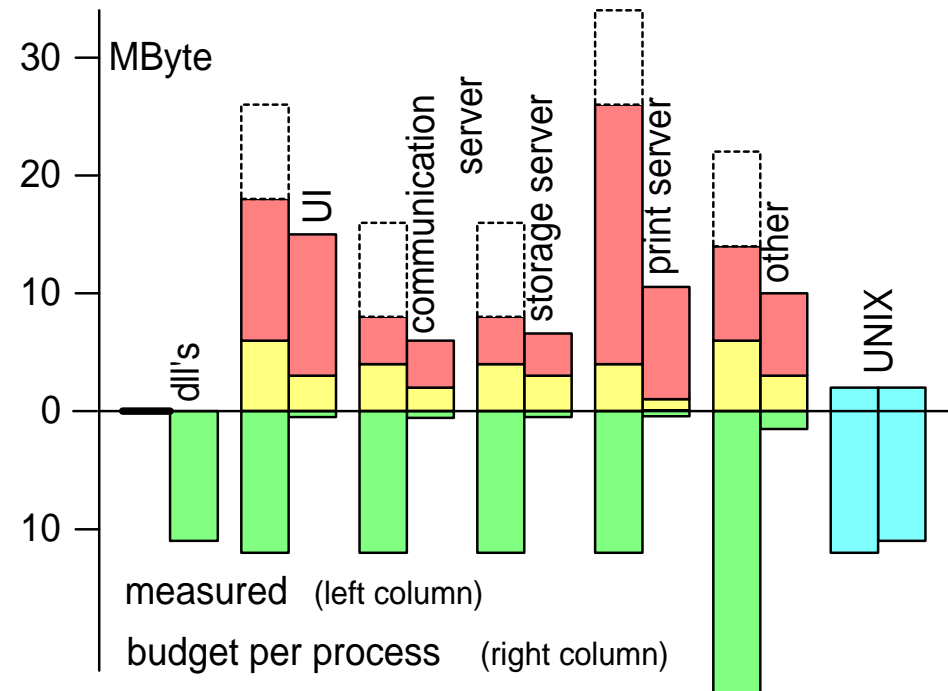
Method: Budget per Process

Budget:

+ measurable

+ fine enough to provide direction

+ coarse enough to be maintainable



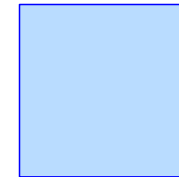
Example of a Memory Budget

<i>memory budget in Mbytes</i>	code	obj data	bulk data	total
shared code	11.0			11.0
User Interface process	0.3	3.0	12.0	15.3
database server	0.3	3.2	3.0	6.5
print server	0.3	1.2	9.0	10.5
optical storage server	0.3	2.0	1.0	3.3
communication server	0.3	2.0	4.0	6.3
UNIX commands	0.3	0.2	0	0.5
compute server	0.3	0.5	6.0	6.8
system monitor	0.3	0.5	0	0.8
application SW total	13.4	12.6	35.0	61.0
UNIX Solaris 2.x				10.0
file cache				3.0
total				74.0

Exercise: Bulk Data Capacity

Memory block
12MByte

How many blocks of
1024 x 1024 8-bits data
can be stored?



How many blocks of
1024 x 1024 16-bits data
can be stored?



Exercise: Object Data Capacity

Object Data
3MByte

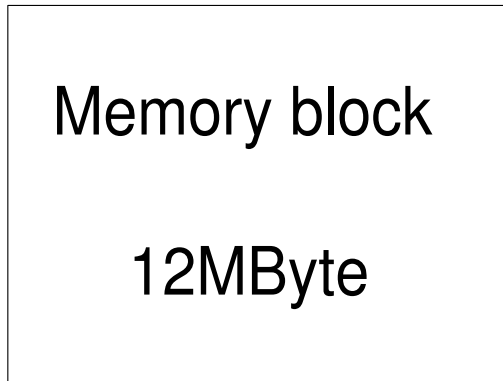
Frequency	Description	Typical size
1	Large objects (e.g. dictionary)	20 kB
20	Medium object, e.g. UI data	200 Bytes
1000	Small object, e.g. image attributes	20 Bytes
Total		

How many objects with this distribution fit in the 3MByte Object data store?

Memory Budget of Easyvision RF R1 and R2

<i>memory budget in Mbytes</i>	code		object data		bulk data		total	
	R1	R2	R1	R2	R1	R2	R1	R2
shared code	6.0	11.0					6.0	11.0
UI process	0.2	0.3	2.0	3.0	12.0	12.0	14.2	15.3
database server	0.2	0.3	4.2	3.2		3.0	4.4	6.5
print server	0.4	0.3	2.2	1.2	7.0	9.0	9.6	10.5
DOR server	0.4	0.3	4.2	2.0	2.0	1.0	6.6	3.3
communication server	1.2	0.3	15.4	2.0	10.0	4.0	26.6	6.3
UNIX commands	0.2	0.3	0.5	0.2			0.7	0.5
compute server		0.3		0.5		6.0		6.8
system monitor		0.3		0.5				0.8
application total	8.6	13.4	28.5	12.6	31.0	35.0	66.1	61.0
UNIX file cache							7.0 3.0	10.0 3.0
total							76.1	74.0

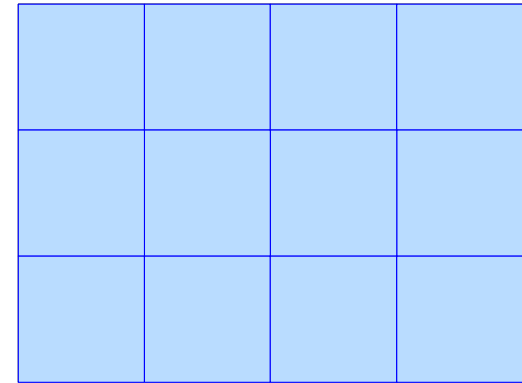
Answer: Bulk Data Capacity



How many blocks of
1024 x 1024 8-bits data
can be stored?



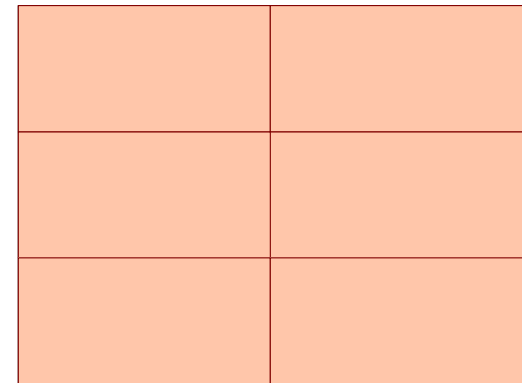
12



How many blocks of
1024 x 1024 16-bits data
can be stored?



6



* Assuming that 8-bit data is stored as 8-bit (char)
Assuming that 16-bit data is stored as 16-bit (short int)

Answer: Object Data Capacity

Object Data
3MByte

Frequency	Description	Typical size	Size * Freq
1	Large objects (e.g. dictionary)	20 kB	20 kB
20	Medium object, e.g. UI data	200 Bytes	4kB
1000	Small object, e.g. image attributes	20 Bytes	20kB
Total			44kB

44kByte fits approximately 68 times in 3MByte
Expect to store at most 68 large objects
(1360 Medium sized objects, 68000 small objects)

Easyvision Memory Design

Fragmentation and consequences

Application caches

Memory design applied

Memory Fragmentation

image 1, 256 kB | image 2, 256 kB | image 3, 256 kB

1. replace image 3 by image 4



image 1, 256 kB |  | image 3, 256 kB

image 1, 256 kB | 4 |  | image 3, 256 kB

2. add image 5

image 1, 256 kB | 4 |  | image 3, 256 kB | image 5, 256 kB

3. replace image 1 by image 6

 | 4 |  | image 3, 256 kB | image 5, 256 kB

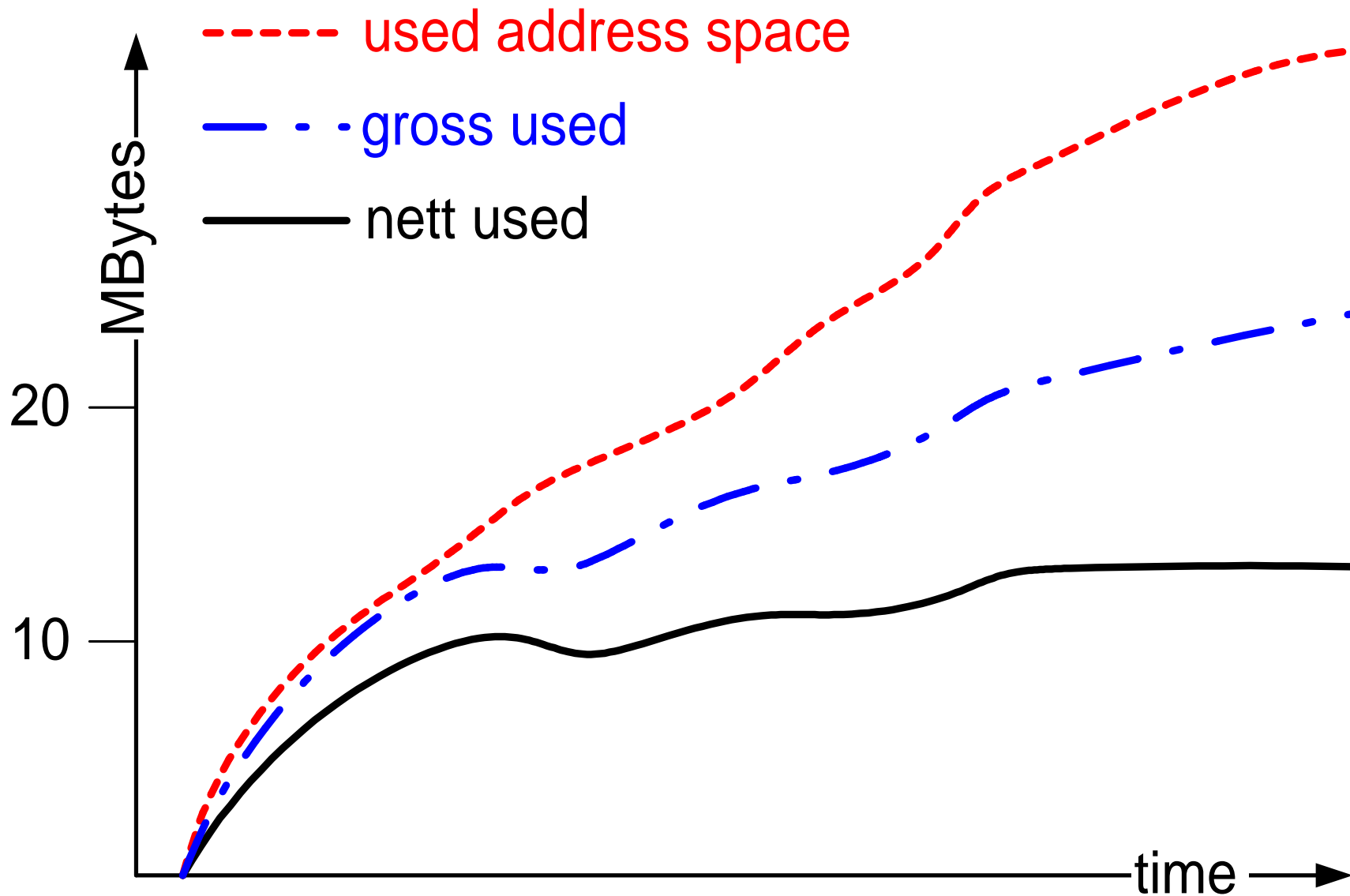
6 |  | 4 |  | image 3, 256 kB | image 5, 256 kB

legend

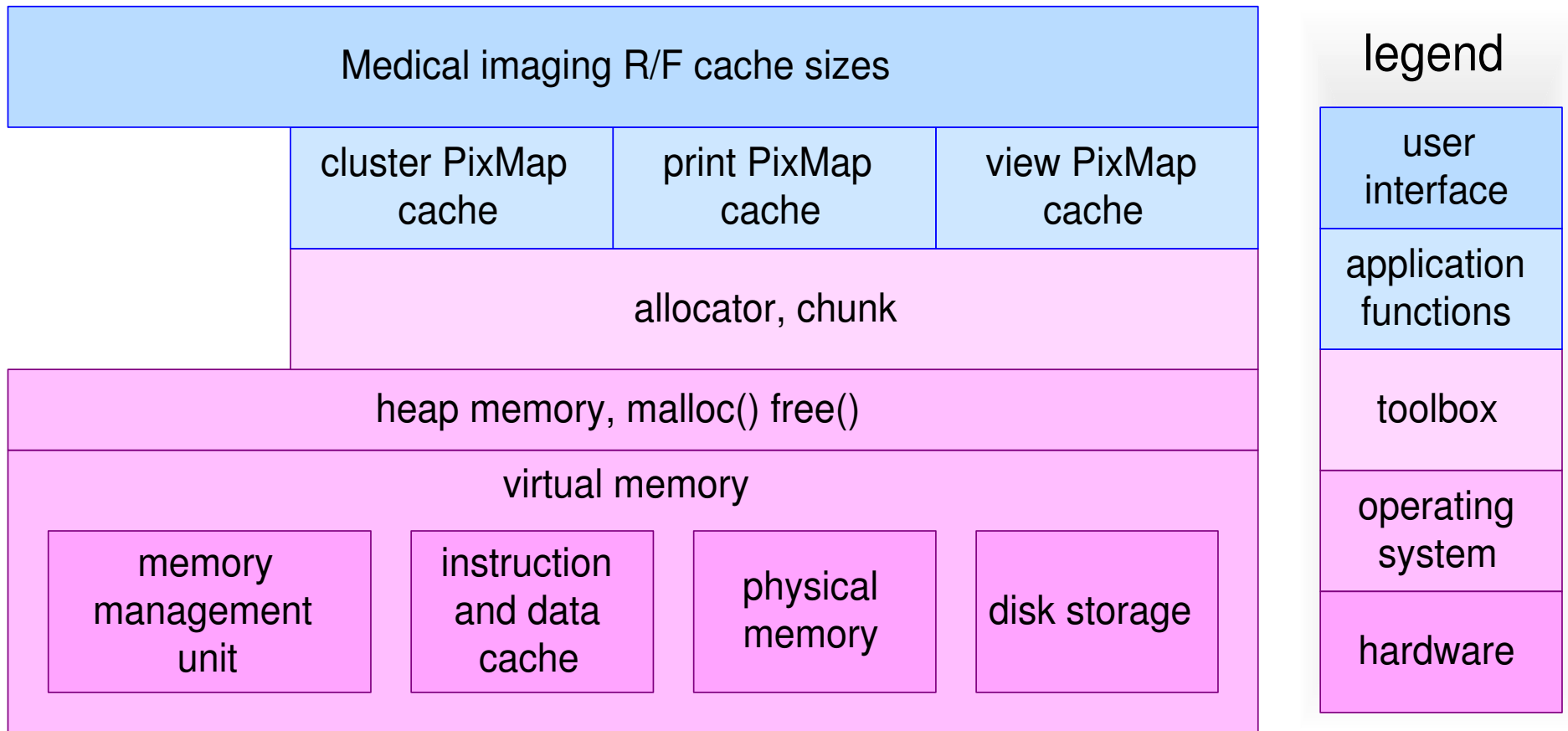
image in use

unused memory

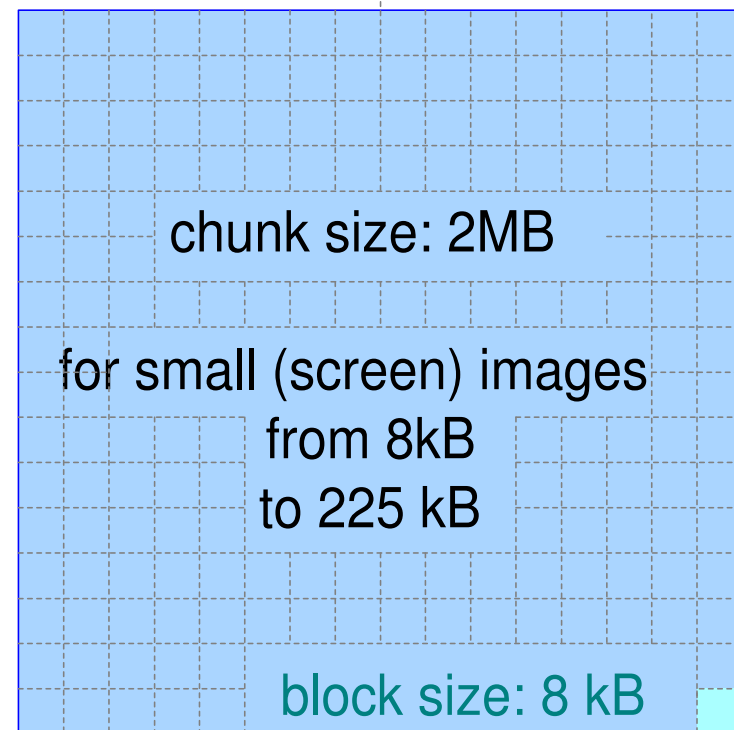
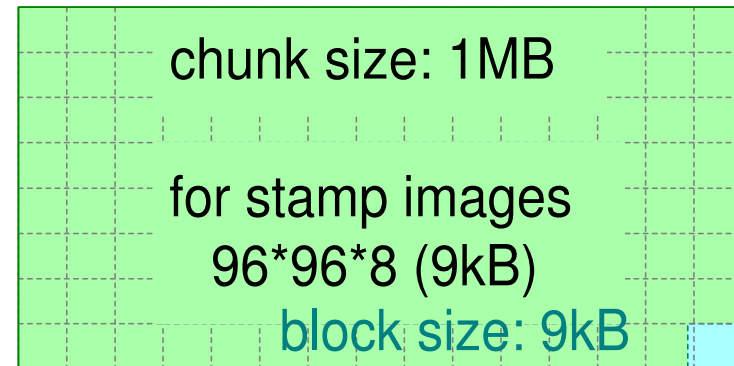
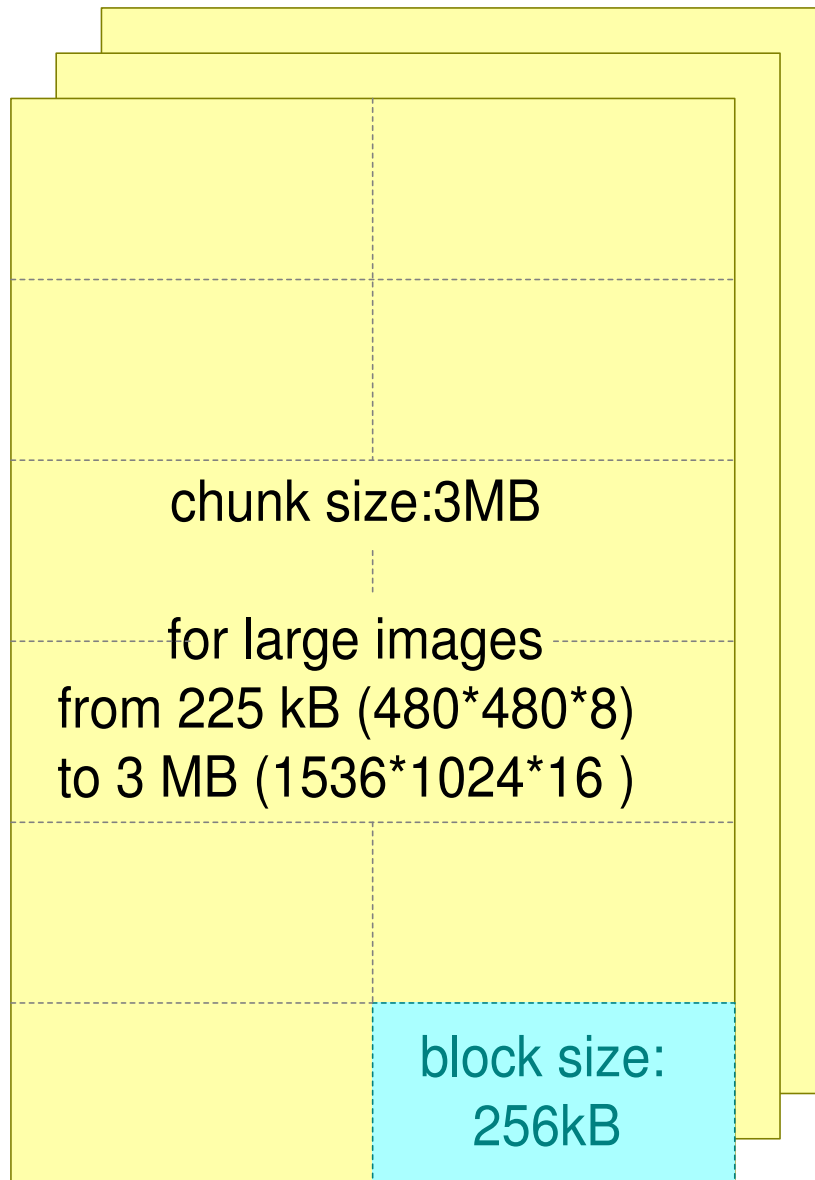
Memory Fragmentation Increase



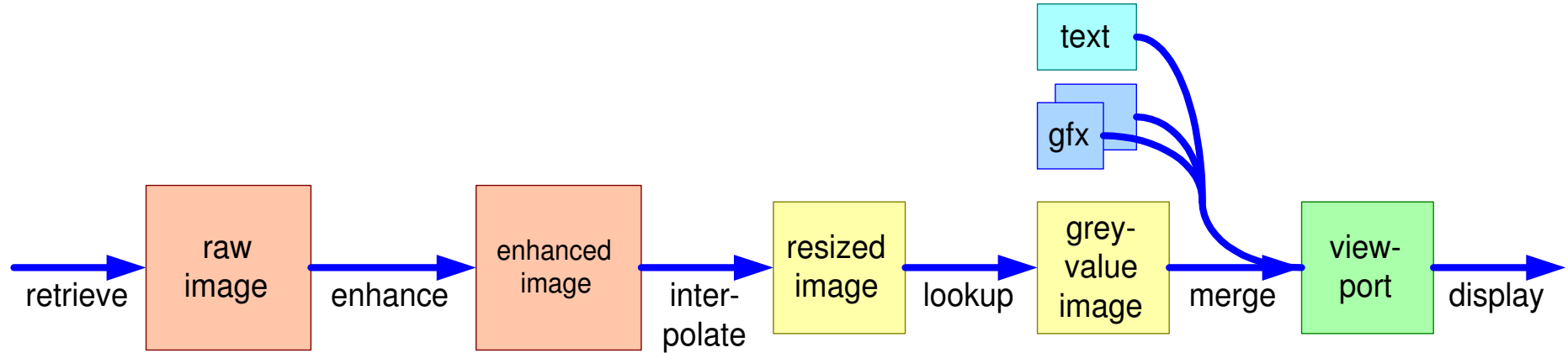
Cache Layers



Bulk Data Memory Management Memory Allocators



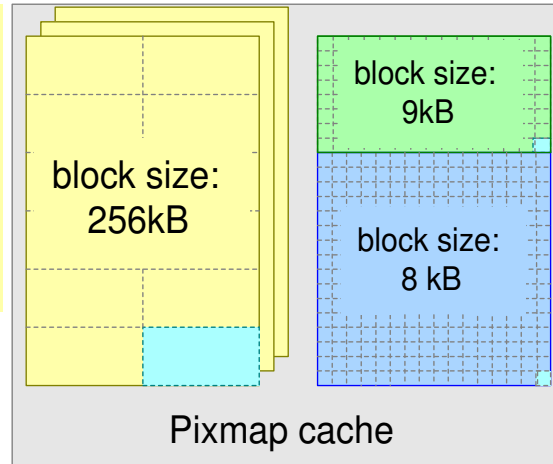
Cached Intermediate Processing Results



Example of Allocator and Cache Use

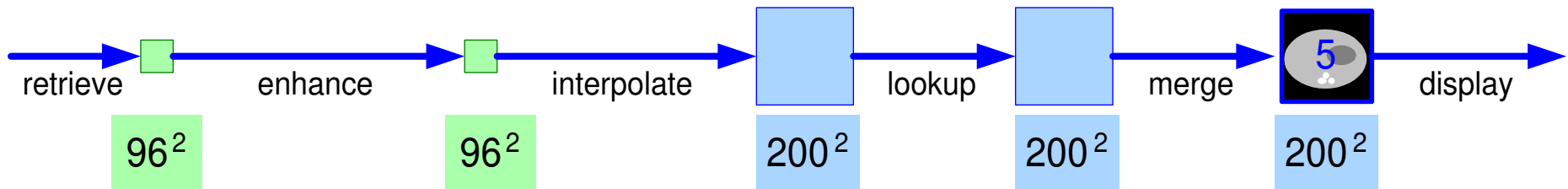
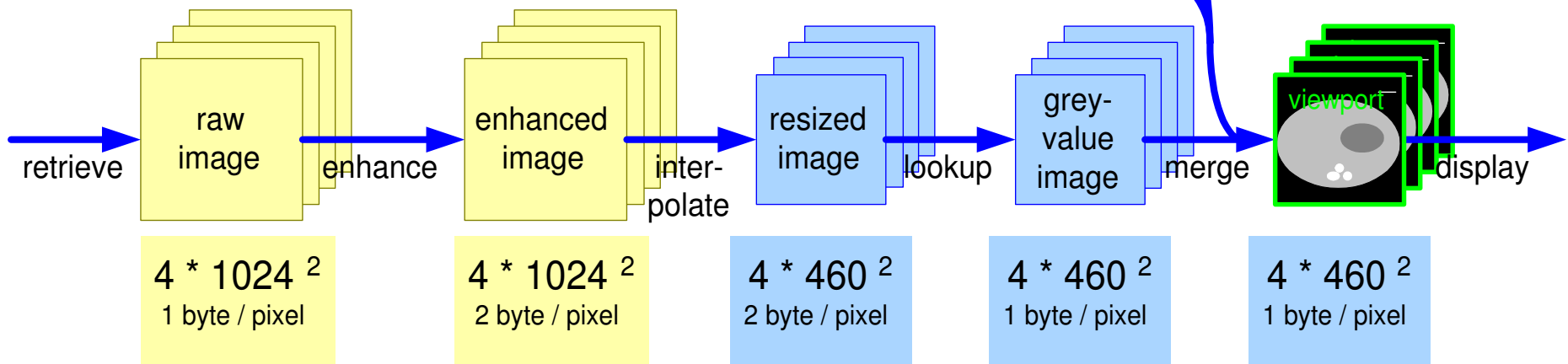
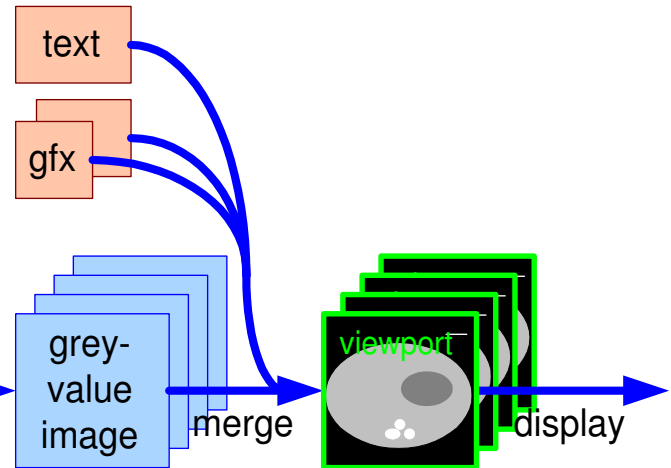
1024² 8 bit image requires
4 256kB blocks

8 1024² images require
48 256kB blocks
12 blocks shortage

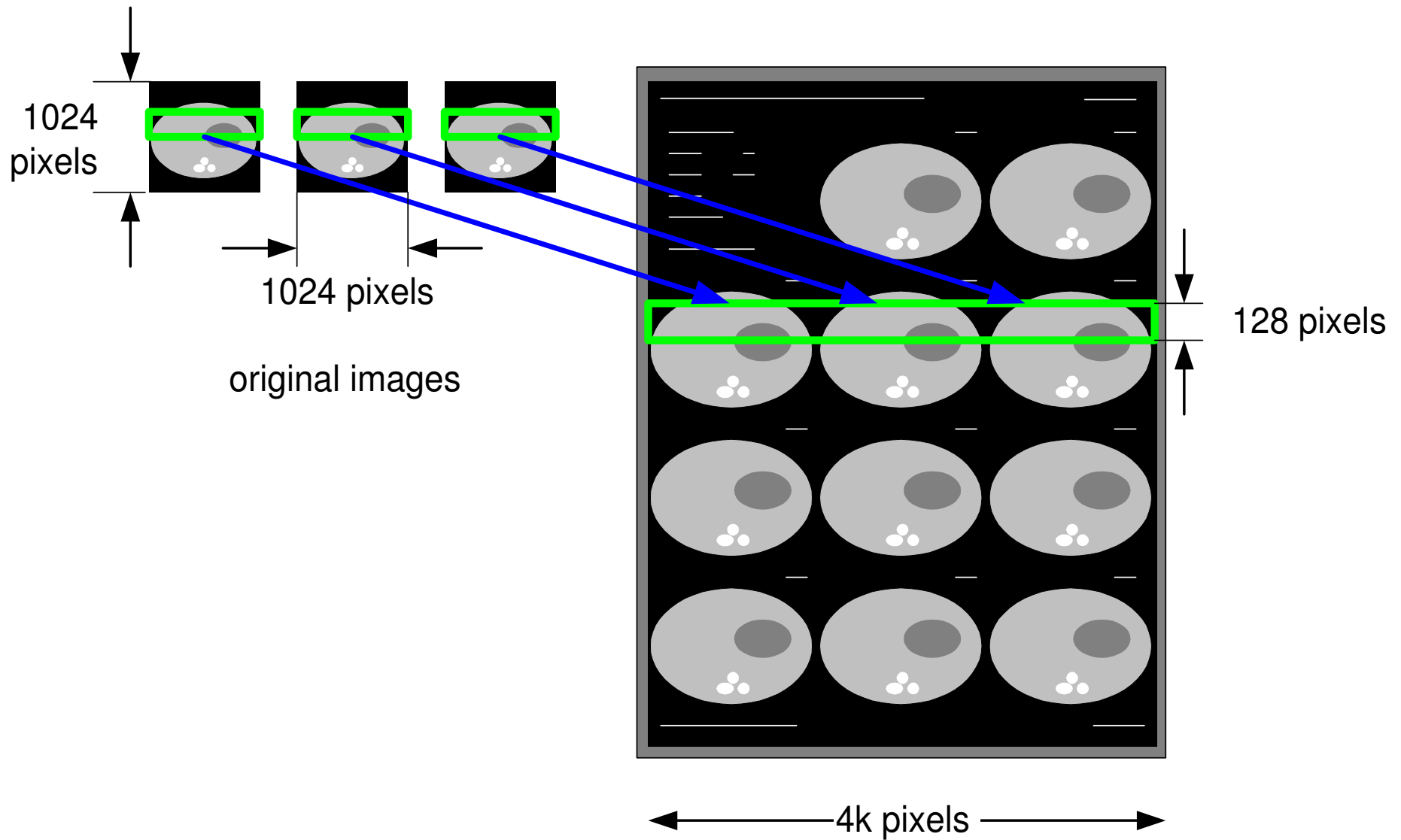


460² image 8 bit requires 27 8kB blocks
200² images require 5 8kb blocks

all screen-size images require
334 8kB blocks, 78 blocks shortage



Print Server is Based on Banding



Easyvision Memory CPU load and performance

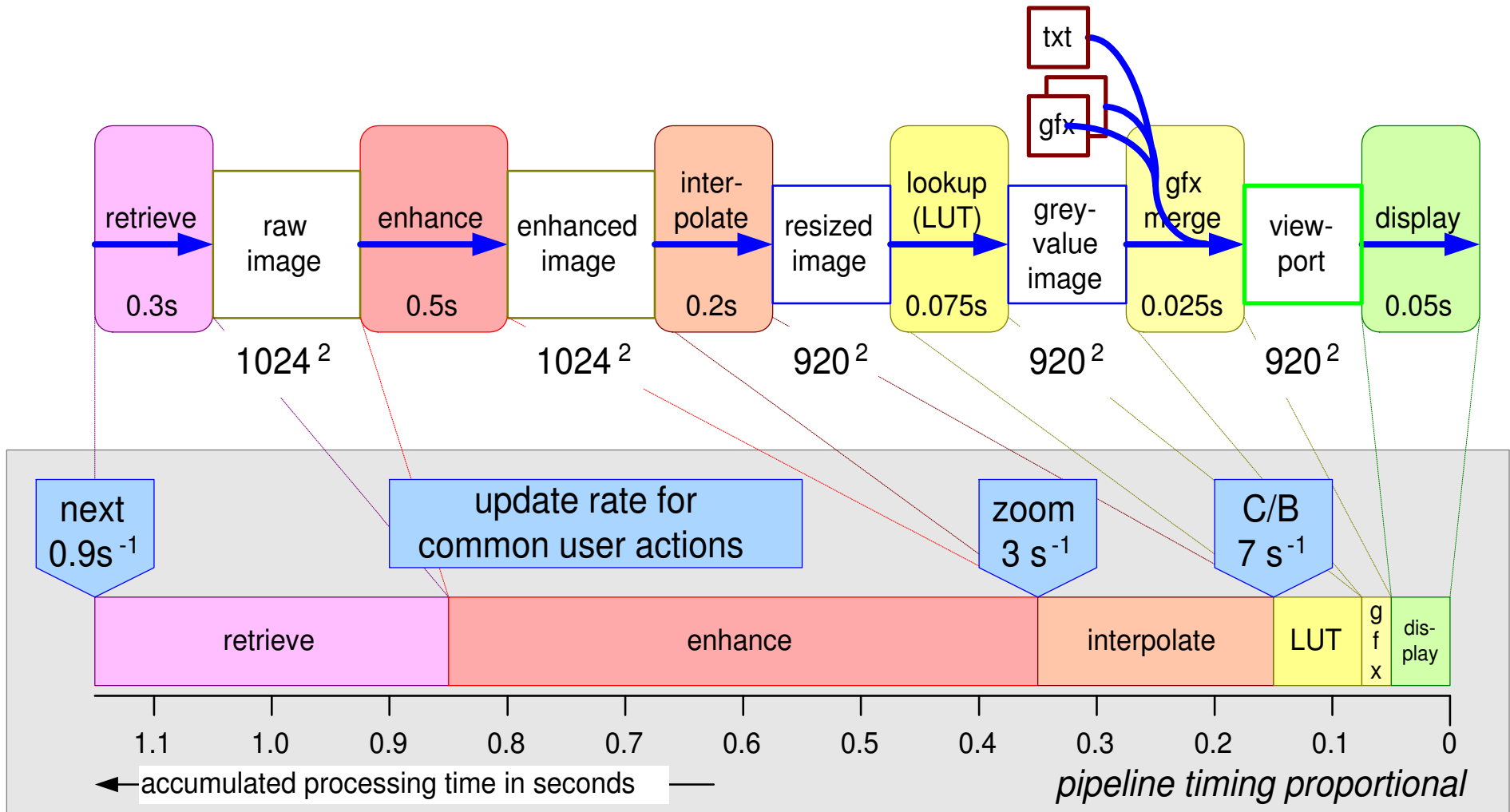
CPU load analysis

response time

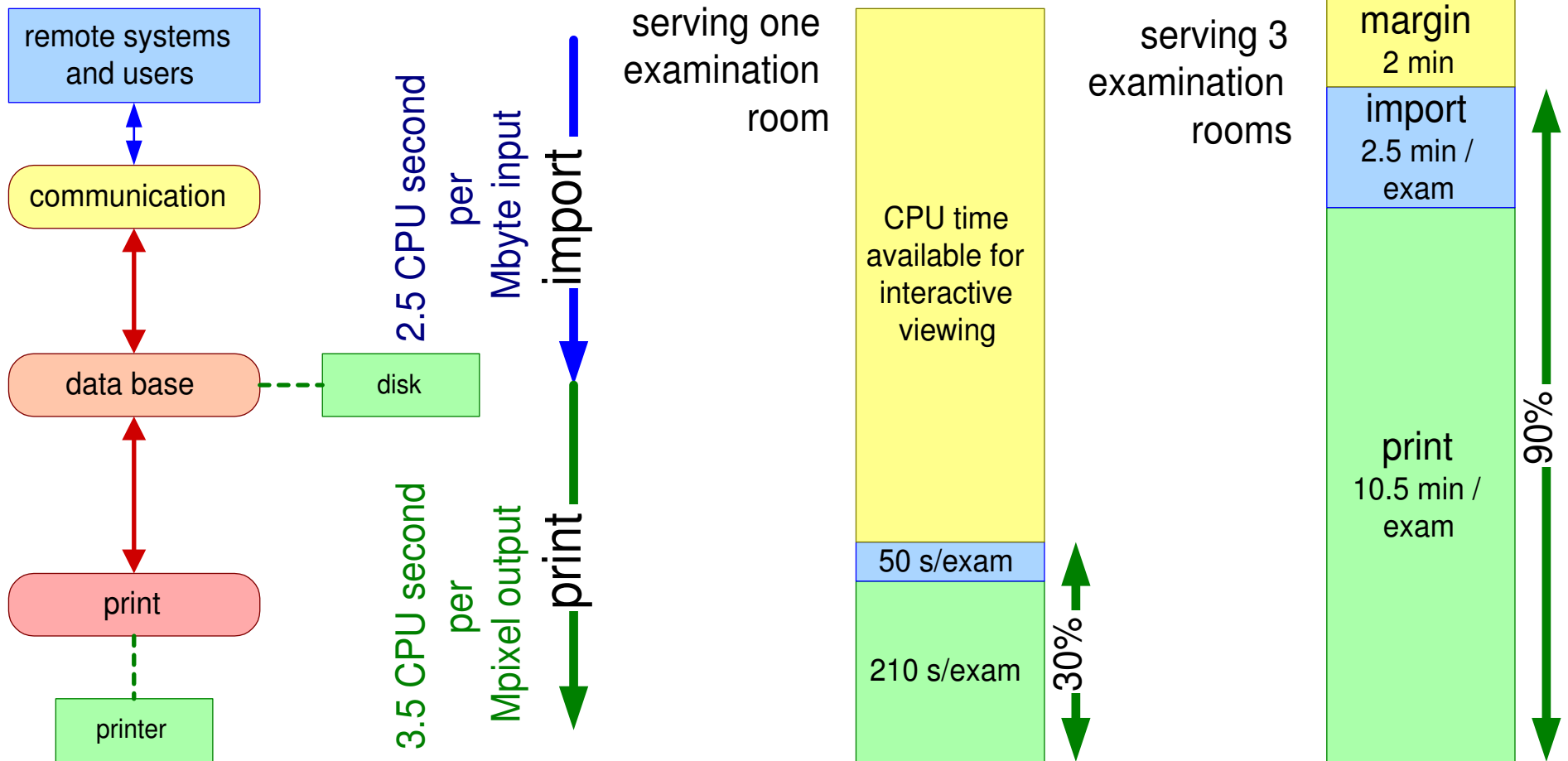
throughput

measurement tools

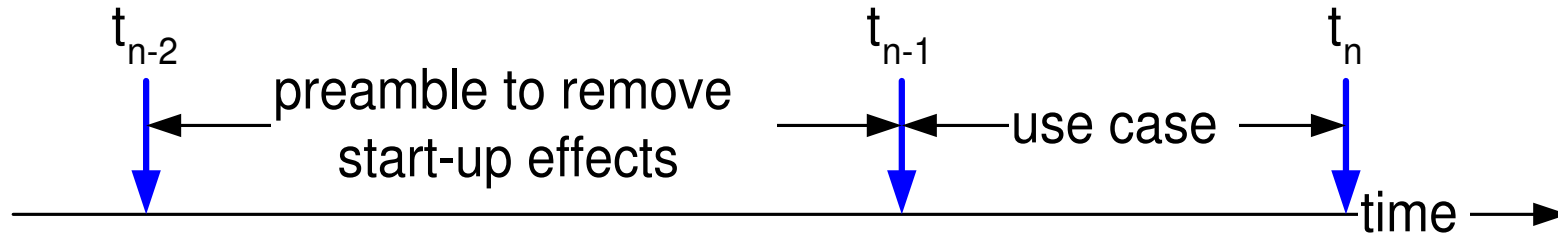
CPU Processing Times and Viewing Responsiveness



Server CPU Load



Resource Measurement Tools



oit

△ object instantiations
heap memory usage

ps
vmstat
kernel resource
stats

kernel CPU time
user CPU time
code memory
virtual memory
paging

heapviewer (visualise fragmentation)

Object Instantiation Tracing

class name	current nr of objects	deleted since t_{n-1}	created since t_{n-1}	heap memory usage
AsynchronousIO	0	-3	+3	
AttributeEntry	237	-1	+5	
BitMap	21	-4	+8	
BoundedFloatingPoint	1034	-3	+22	
BoundedInteger	684	-1	+9	
BTreeNode1	200	-3	+3	[819200]
BulkData	25	0	1	[8388608]
ButtonGadget	34	0	2	
ButtonStack	12	0	1	
ByteArray	156	-4	+12	[13252]

Overview of Benchmarks and Other Measurement Tools

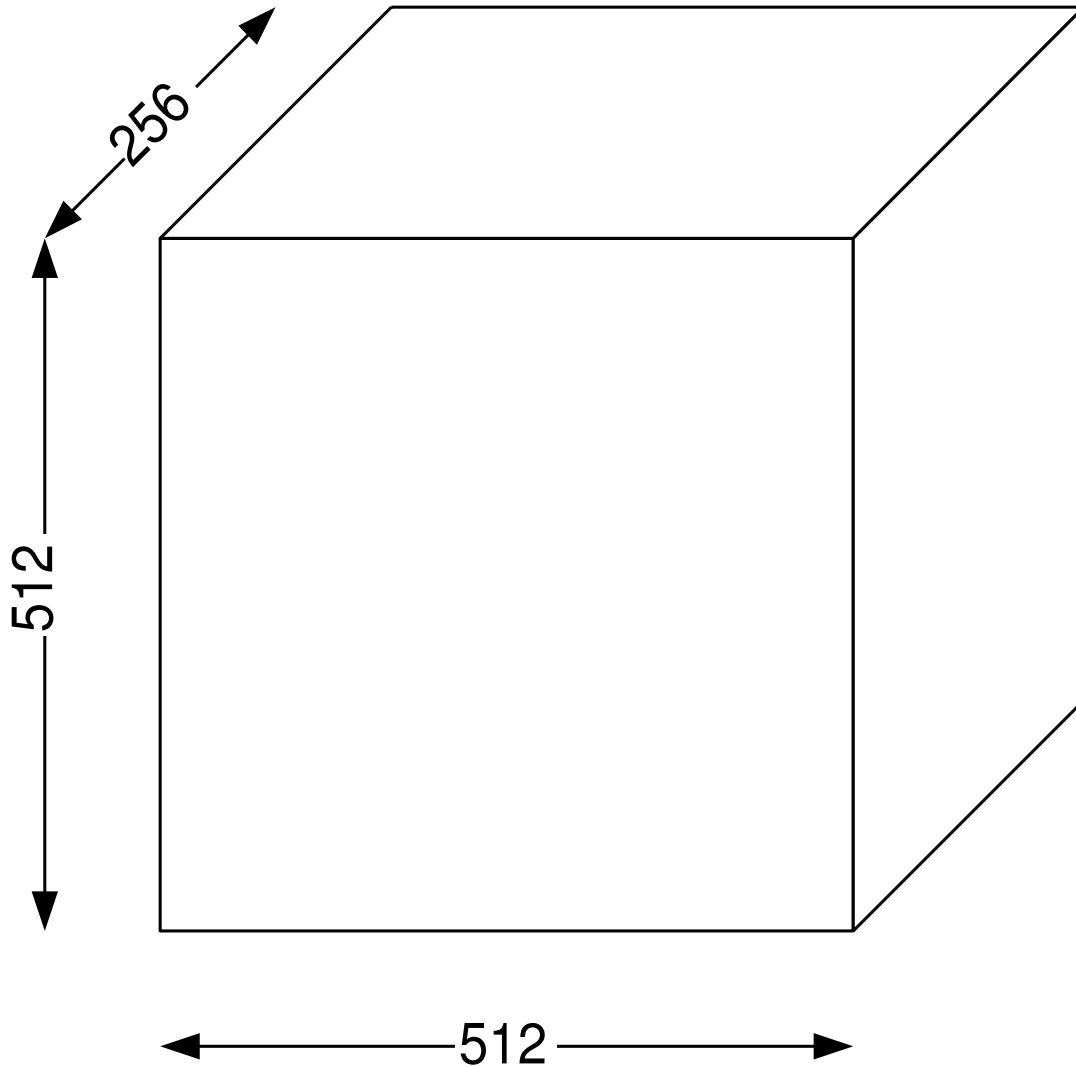
	test / benchmark	what, why	accuracy	when
<i>public</i>	SpecInt (by suppliers)	CPU integer	coarse	new hardware
	Byte benchmark	computer platform performance OS, shell, file I/O	coarse	new hardware new OS release
<i>self made</i>	file I/O	file I/O throughput	medium	new hardware
	image processing	CPU, cache, memory as function of image, pixel size	accurate	new hardware
	Objective-C overhead	method call overhead memory overhead	accurate	initial
	socket, network	throughput CPU overhead	accurate	ad hoc
	data base	transaction overhead query behaviour	accurate	ad hoc
	load test	throughput, CPU, memory	accurate	regression

MRI Volume Reconstruction and Viewing

Usage patterns as impact on performance

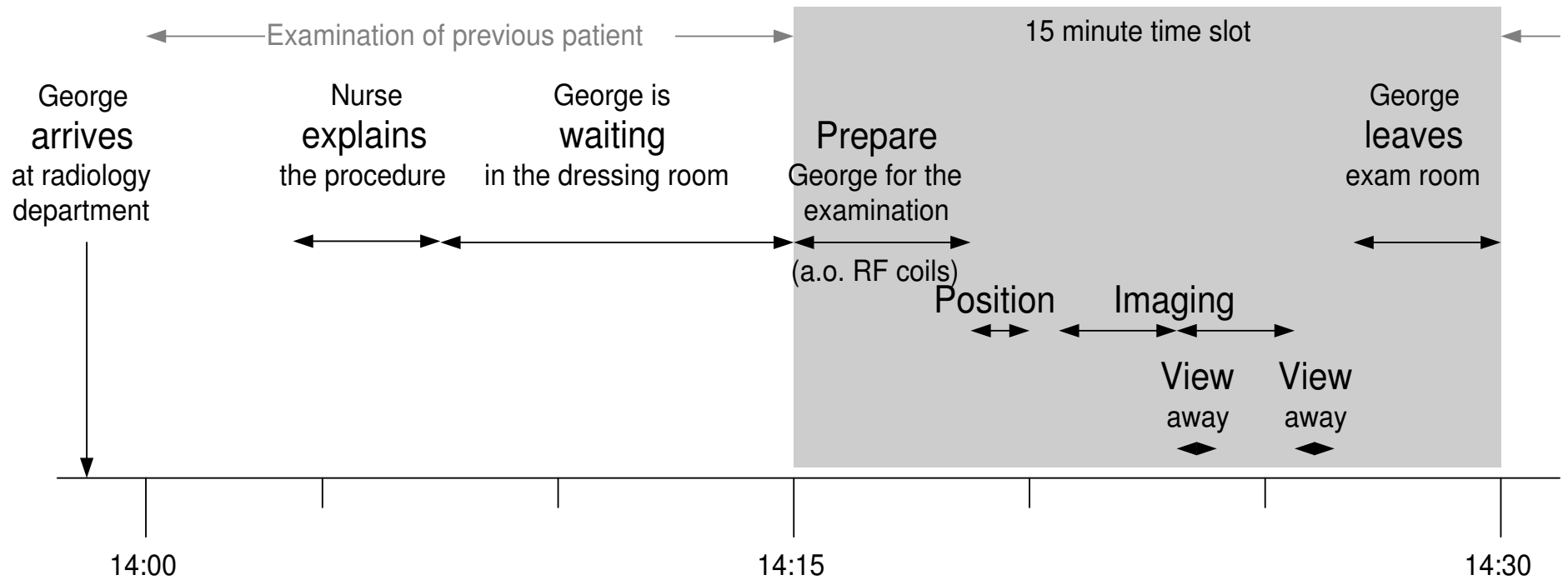
Resource model and requirements identification for usage patterns

Volume Acquisition and Reconstruction

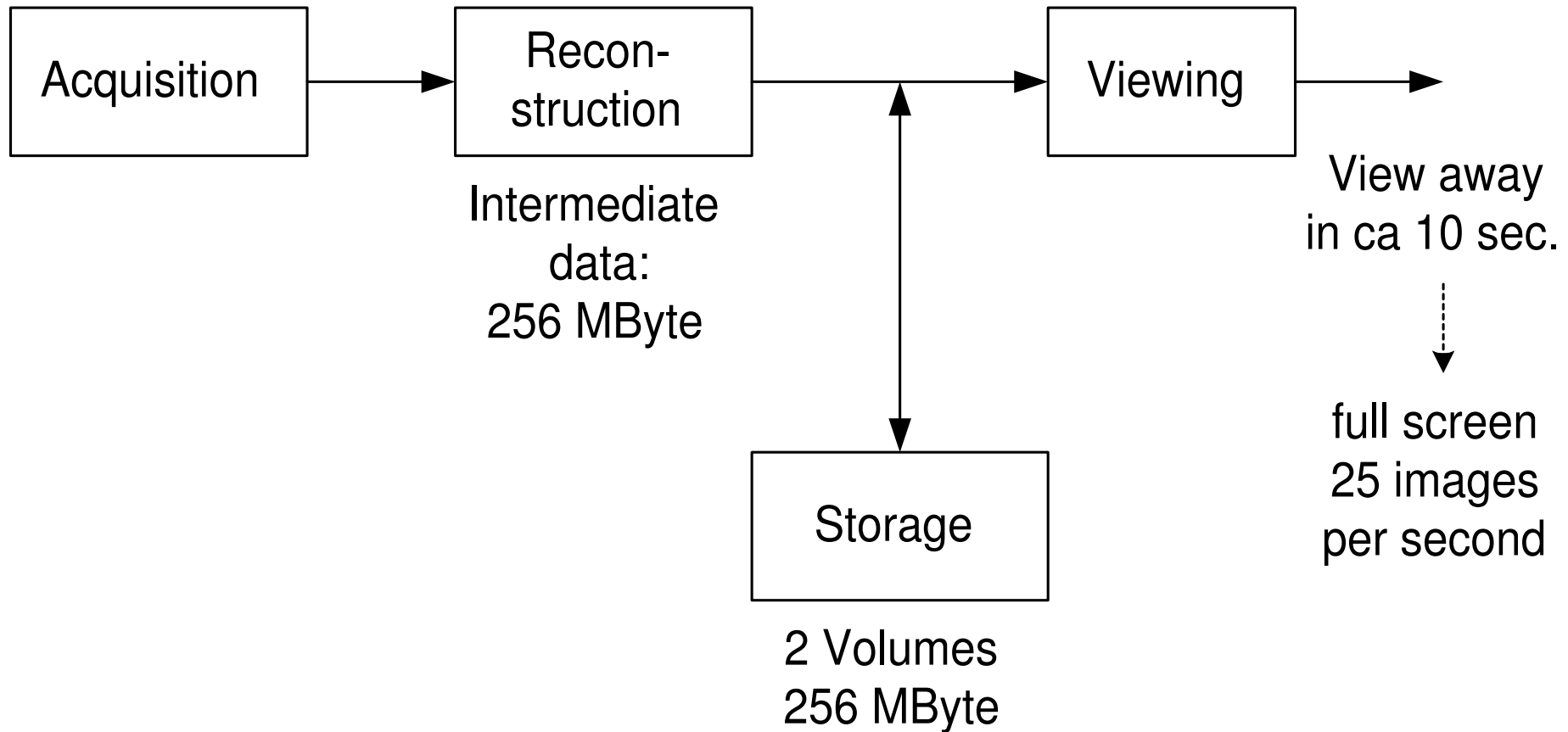


$$\begin{array}{l} \text{Data in bytes} = \\ 2 * 512 * 512 * 256 * 2 = \\ \text{Volumes} \quad \times \quad \times \quad \times \quad \text{bytes per pixel} \\ \\ 256 \text{ MBytes} \\ \\ \text{in } 2 * 2 \text{ minutes} = \\ 240 \text{ seconds} \end{array}$$

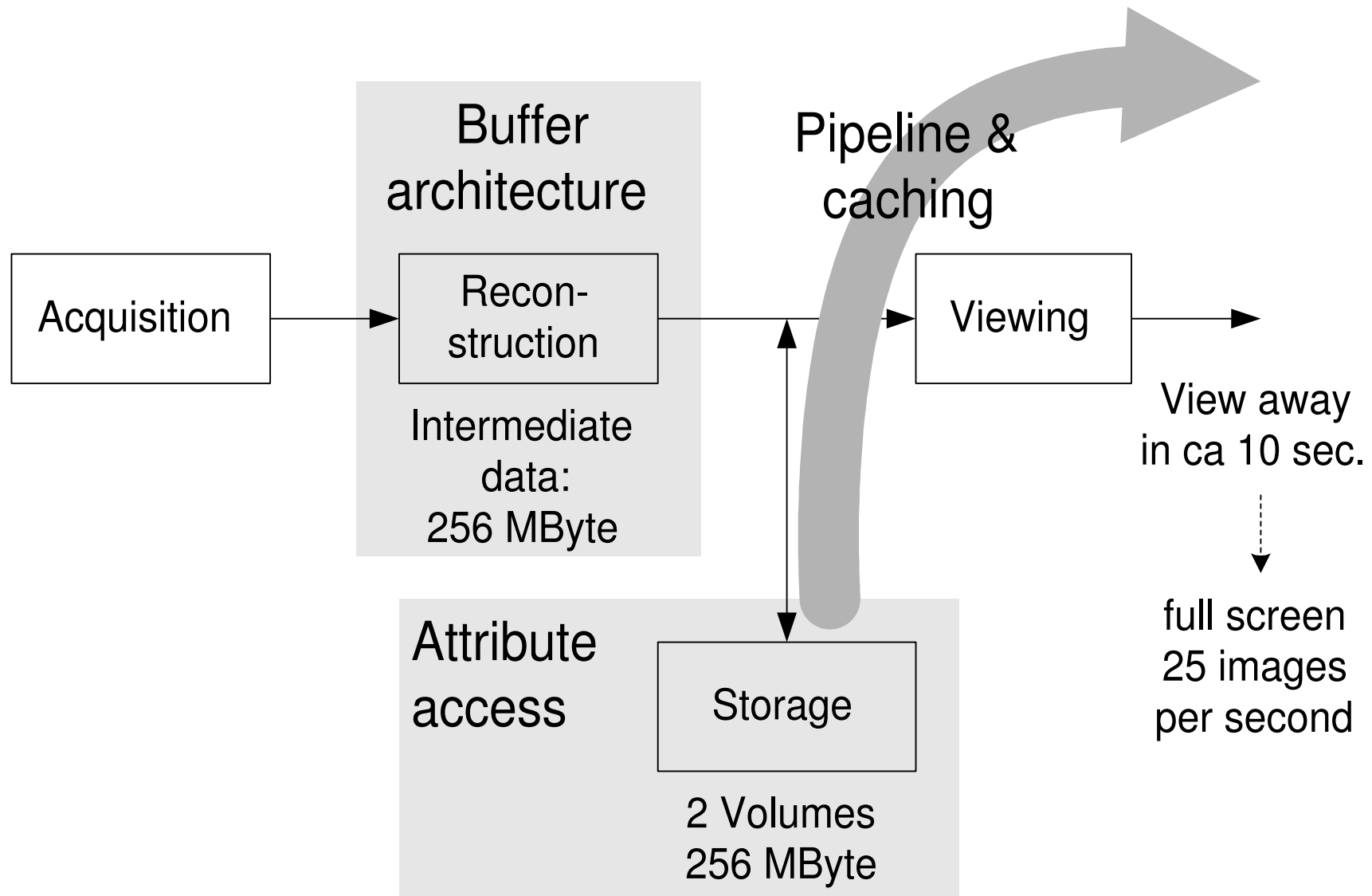
Performance Requirements



Resource Model



Critical Resources



MRI Volume Reconstruction and Viewing

Operational usage pattern drives (implicit/explicit) system performance requirements

Resource / cost trade-off must support operational usage patterns

Mobile Display Appliances

Modelling external environment

End-to-end performance

Allocation choices

Mobile Display Appliance

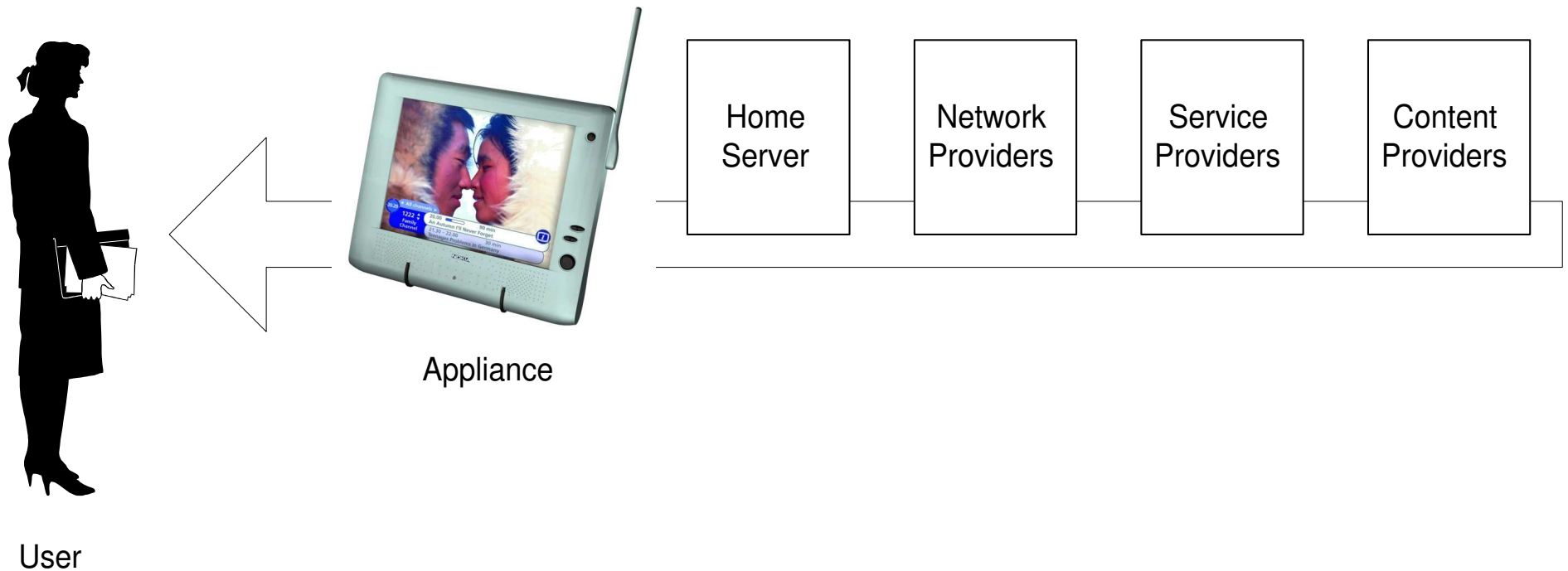


Mediascreen



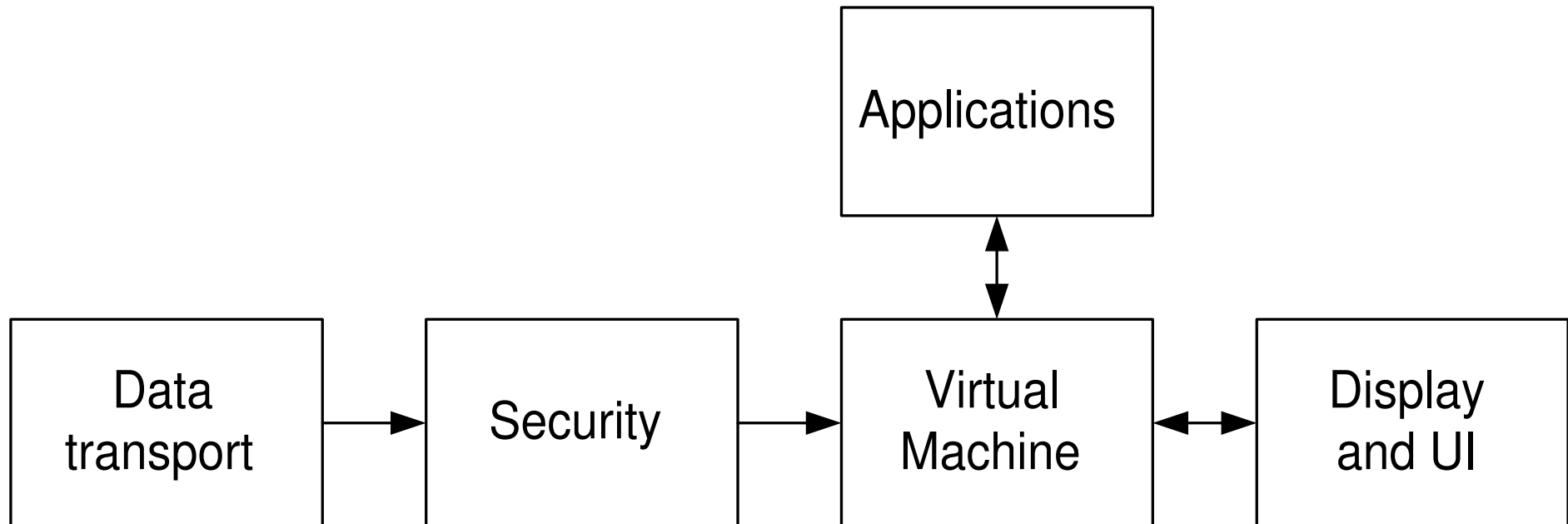
Original pictures from Nokia

User Access Point to a Long Foodchain



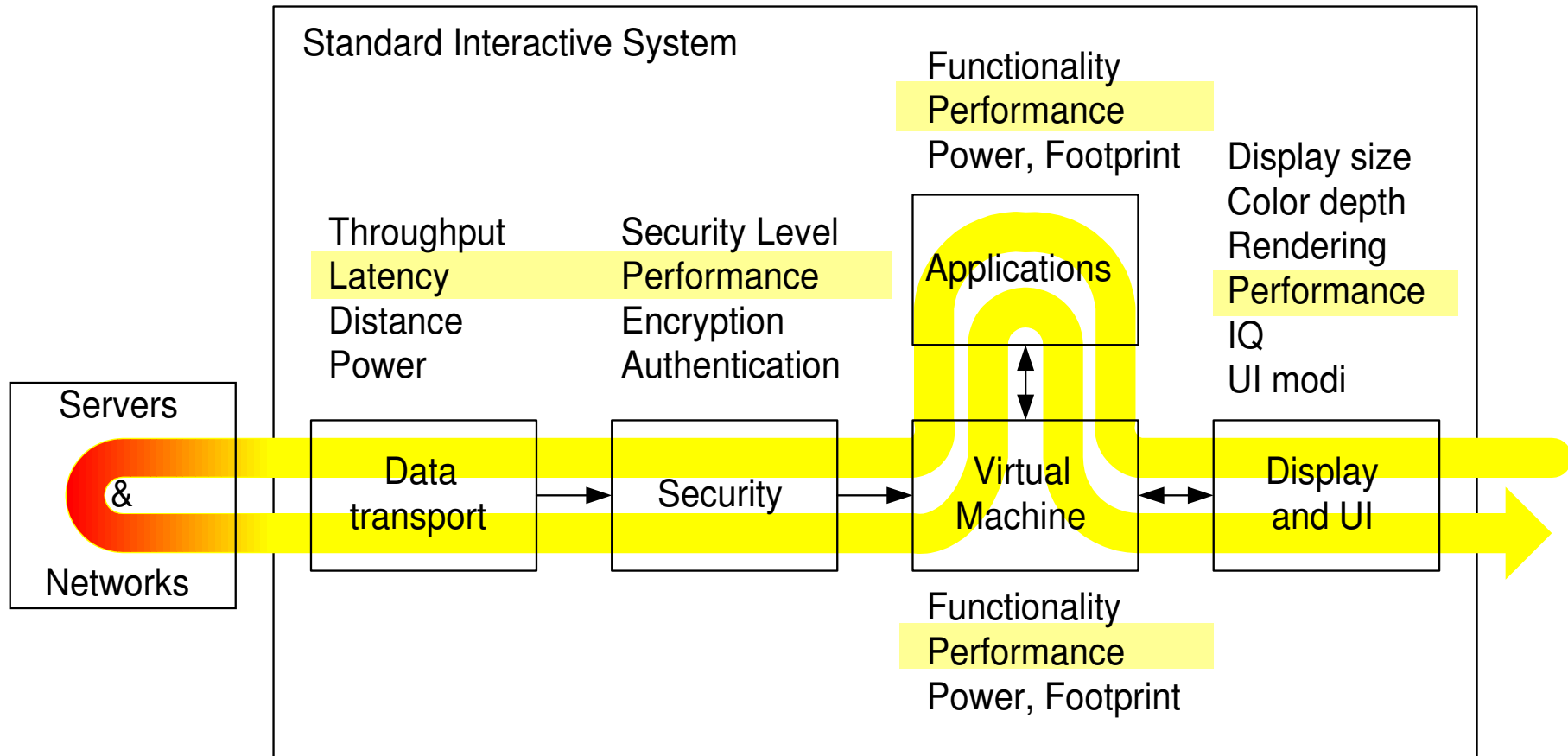
The "SMART" World of the Design

Standard Interactive System



free after Nick Thorne, Philips Semiconductors,
Systems Laboratory Southampton UK,
as presented at PSAVAT April 2001

Specifiable Characteristics

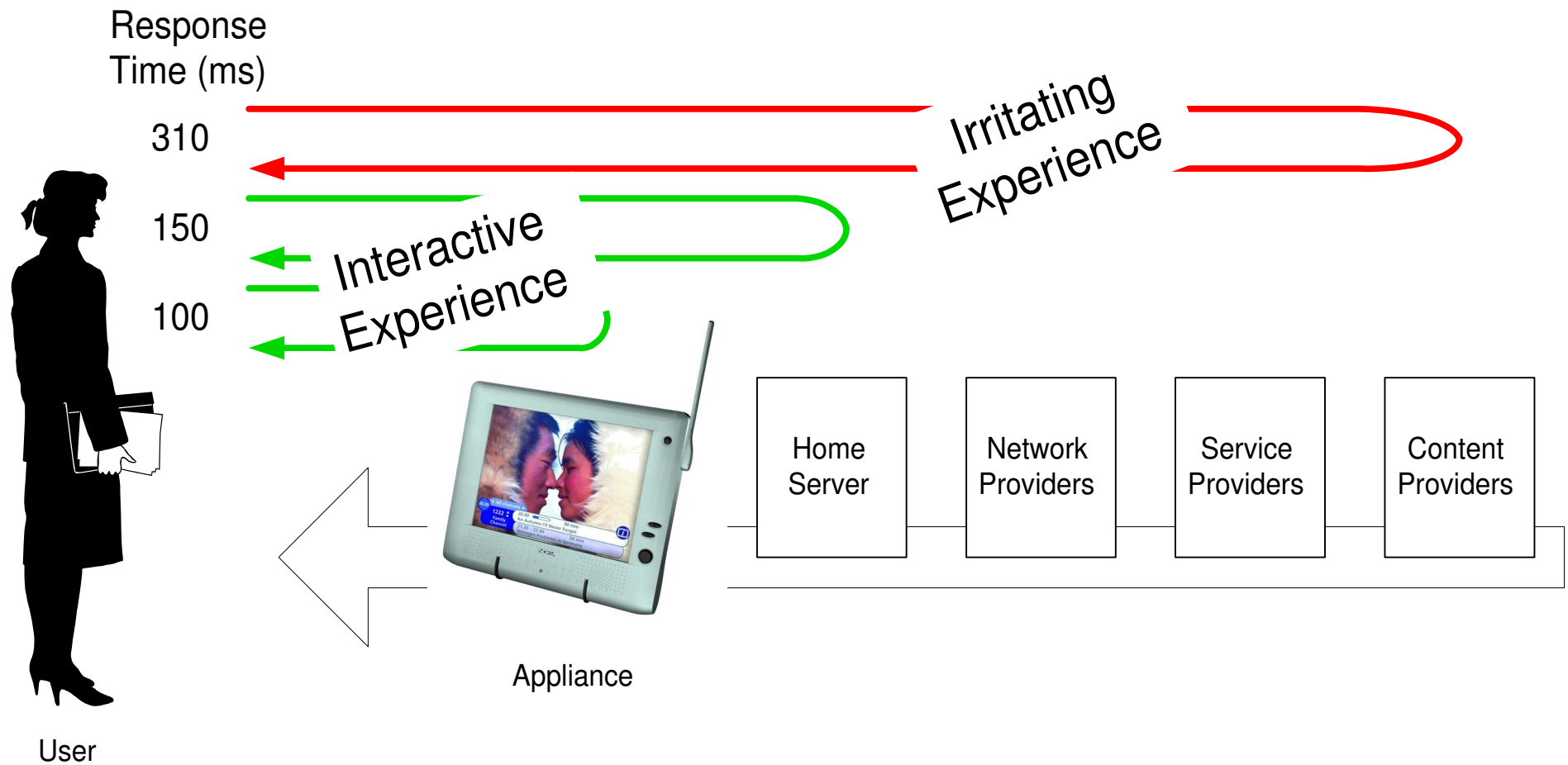


Response Time: Latency Budget

times in milliseconds	Message Latency	Response Time
Appliance	40	100
Data transport	10	20
Security	10	20
Virtual Machine	10	20
Application	10	30
Graphics and UI	0	10
Home Network	20	50
Home Server	10	30
Network contention	10	20
Provider Infrastructure	50	160
Last-Mile network	10	20
Backbone network	20	40
Service server	10	50
Content server	10	50
Total	110	310
User need		200

All numbers are imaginary and for illustration purposes only

Interaction or Irritation?



Mobile Display Appliances

Modelling external environment: make assumptions

End-to-end performance:

large part of performance budget is not controlled

User perceived performance determines function allocation

The ASP TM course is partially derived from the EXARCH course developed at *Philips CTT* by *Ton Kostelijk* and *Gerrit Muller* .

Extensions and additional slides have been developed at *ESI* by *Teun Hendriks* , *Roland Mathijssen* and *Gerrit Muller* .