Runtime resource assurance and adaptation with Qinna framework : a case study

Real Time Systems, October, 20th 2008

Laure Gonnord - Jean-Philippe Babau

University of Lyon - CITI/INSA
France

Context

- Component-based applications.
- Non critical applications : multimedia, . . .
- Restriction to resource management.

Developping resource-aware applications

The developer needs tools to :

- easily add/remove functionalities (compilation/runtime) ;
- adapt functionalities to resource (degraded modes) ;
- evaluate the software performance (QoS).

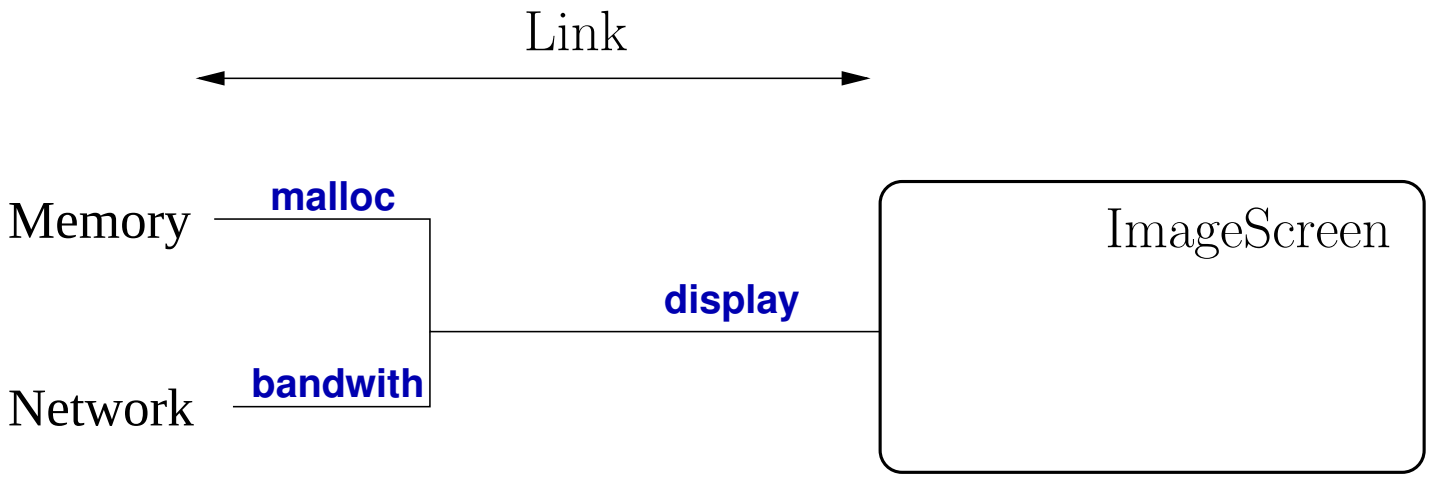
Qinna, a framework to manage QoS at runtime

[Tournier/Babau-CBSE2005]. Main characteristics :

- Component based : application itself + resource + additional ones.
- Software **architecture** : rules, algorithms.
- Services can be degraded : **implementation levels** (IL).
- Automatic management of QoS at **runtime**.
- An implementation in C++.

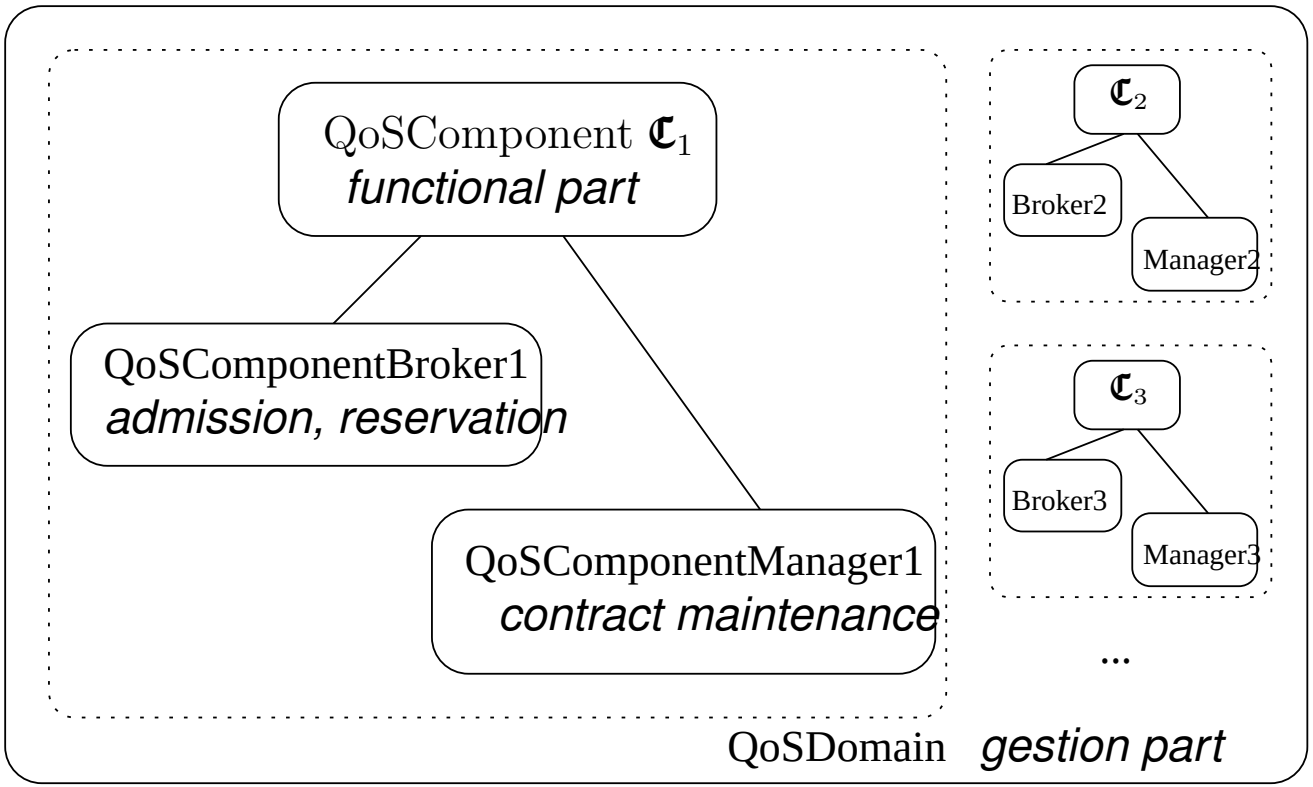
Qinna - 2

- Components provide **services**, which may require other services.



- The links between implementations levels are encoded by the developer (tables).
- Alls demands are asked before : contract $\Rightarrow$ all ILs are fixed in the call tree.

Qinna - 3 - Generic components



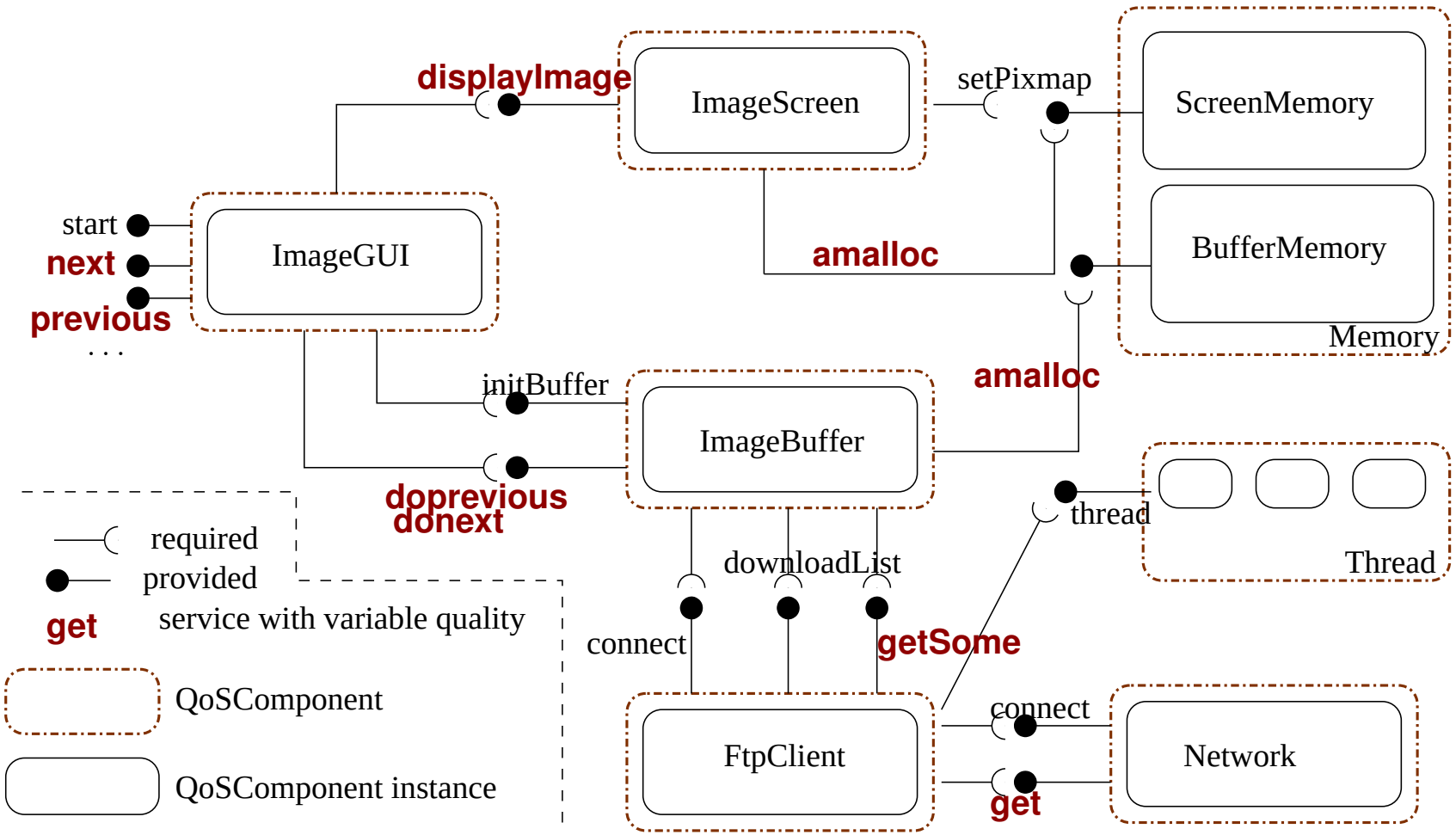
- Automatic management of contracts . . .
- Automatic maintenance of **numerical constraints** (Memory, Network. . .)

Case study : description

- A remote image viewer with ftp connection.
- Adaptation of download and display of the images to :
  - Memory.
  - Bandwidth.
- Evaluation of different policies.

Case study : qinna-isation - 1

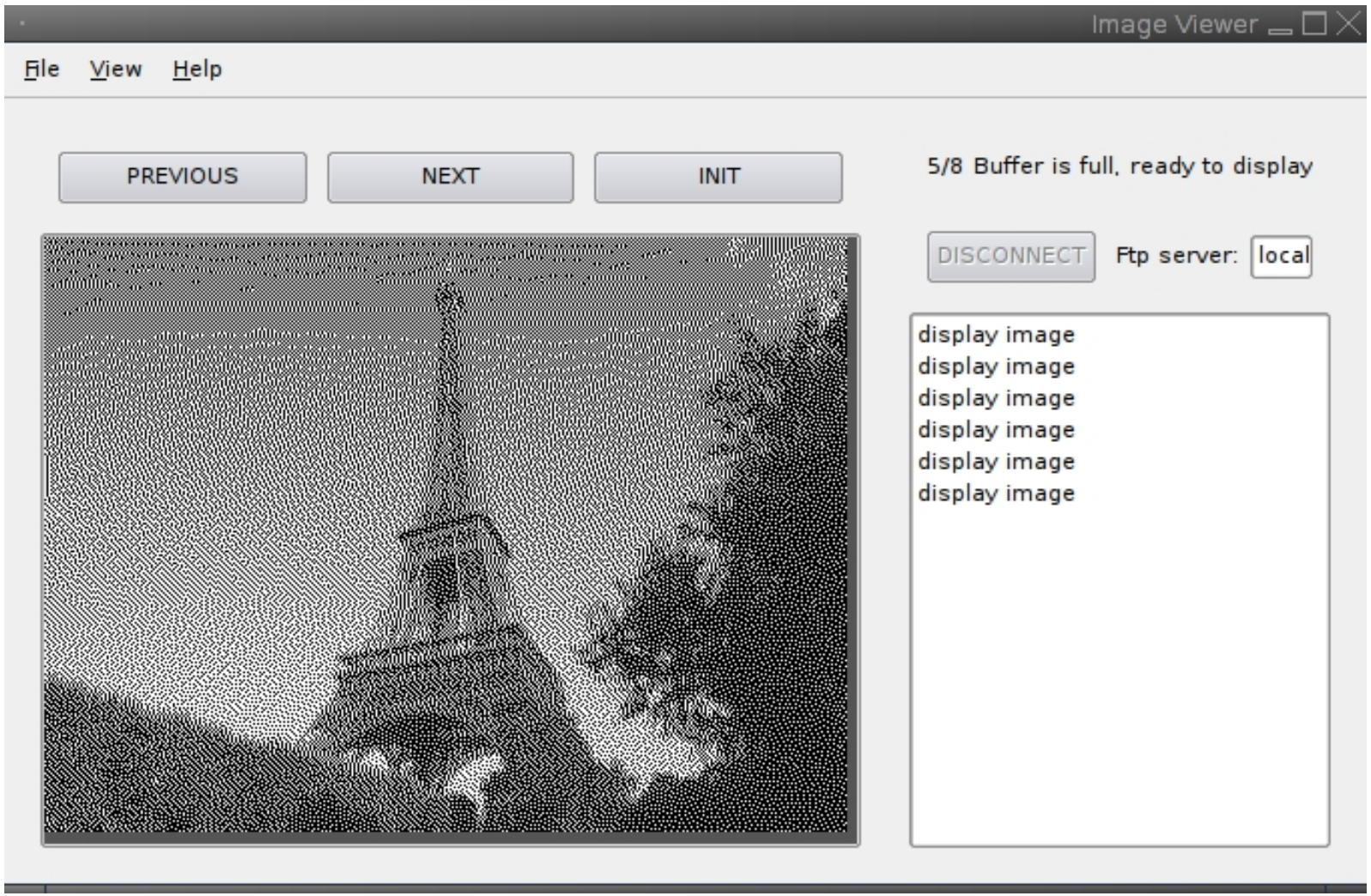
Step 1 : identify the variable services.



Case study : qinna-isation - 1

- Step 2 : Create Qinna components.
- Step 3 : Encode linking constraints
- Step 4 : Encode numerical resource constraints
- Step 5 : Initialise and use Qinna !

Case study : evaluation



Conclusion

Qinna's advantages

- ☺ Séparation of concerns (components).
- ☺ Links between services through tables.
- ☺ Generic negociation algorirthms/components.
- ☺ Distinction between class/instance/service.
- ☺ Distinction between numerical constraints and linking constraints.
- ☺ Effectivity of the method.

But...

- ☹ No discovery of the « best » IL vector *with respect to* some criteria.
- ☹ Overhead too clostly for the moment !

- Future work

Thank you ! !