- Seminar reports and projects:

- Report options:

  - option 1: **general report**

  - option 2: report about the on-the-shelf **hardware** for the main controller with explicit cost/performance comparison

  - option 3: report about the on-the-shelf OS for the main controller with particular reference to the **Alix** and the **RaspBerry Pi** platforms

- Seminar reports and projects:
- Mini project options:
  - web chat application using **Jetty 6** or **Grizzly 2**
  - experimental evaluation of the **EDS bus speed**
  - implementation of a **basic amperometer**
- For the main project please refer to the course website
- for further information: cerocchi@dis.uniroma1.it

- Seminar Schedule [slides available at www.dis.uniroma1.it/~cerocchi]:
  - Lecture 1 - concurrency in home automation systems
  - Lecture 2 - Introduction to home automation system
  - Lecture 3a - Control and Interfaces
    - how to choose the right embedded device
    - how to choose the right software
    - proxy and semantic repository: how to decouple the control layer from the actuation layer
    - Interfaces:
      - buttons
      - touch interfaces
      - how to implement the push notification in the browser: comet technology
      - Eye Tracker
      - Voice Recognition
      - BCI: the brain computer interface
  - Lecture 3b - A real case-study EDS based

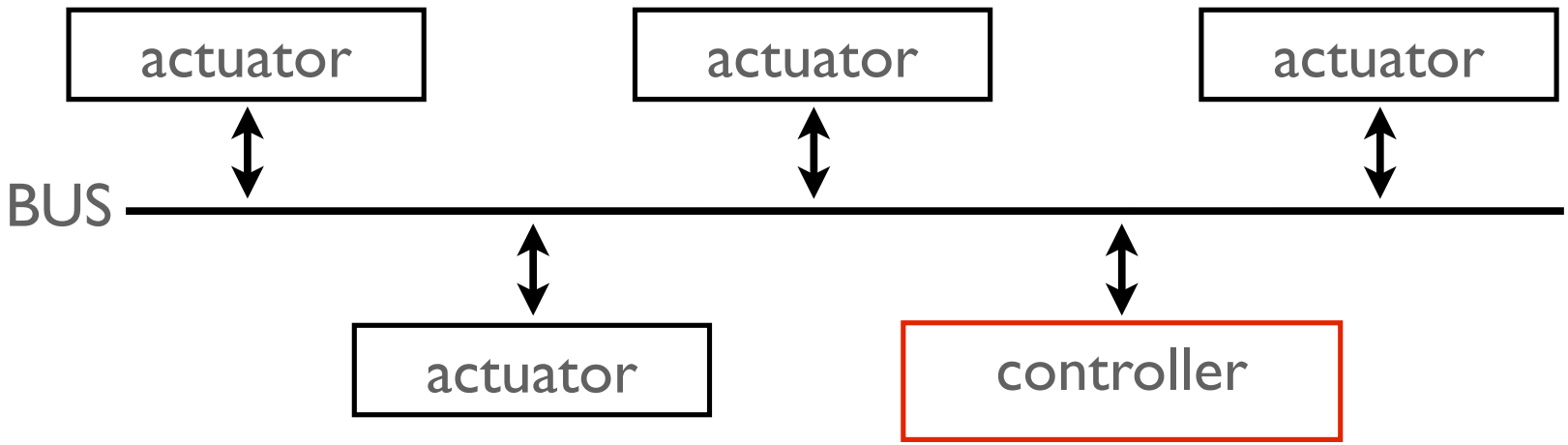
- The control layer

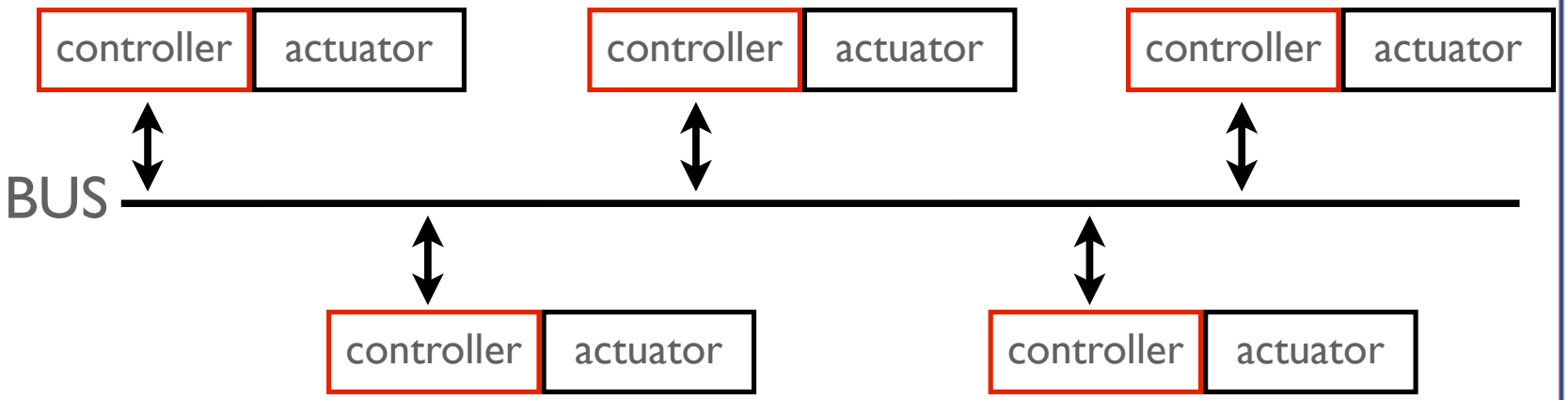
- the control layer is the core of an home automation system: it is the responsible of the decoupling between the **control** and the **actuation**

- it can be both **firmware** and both **software**

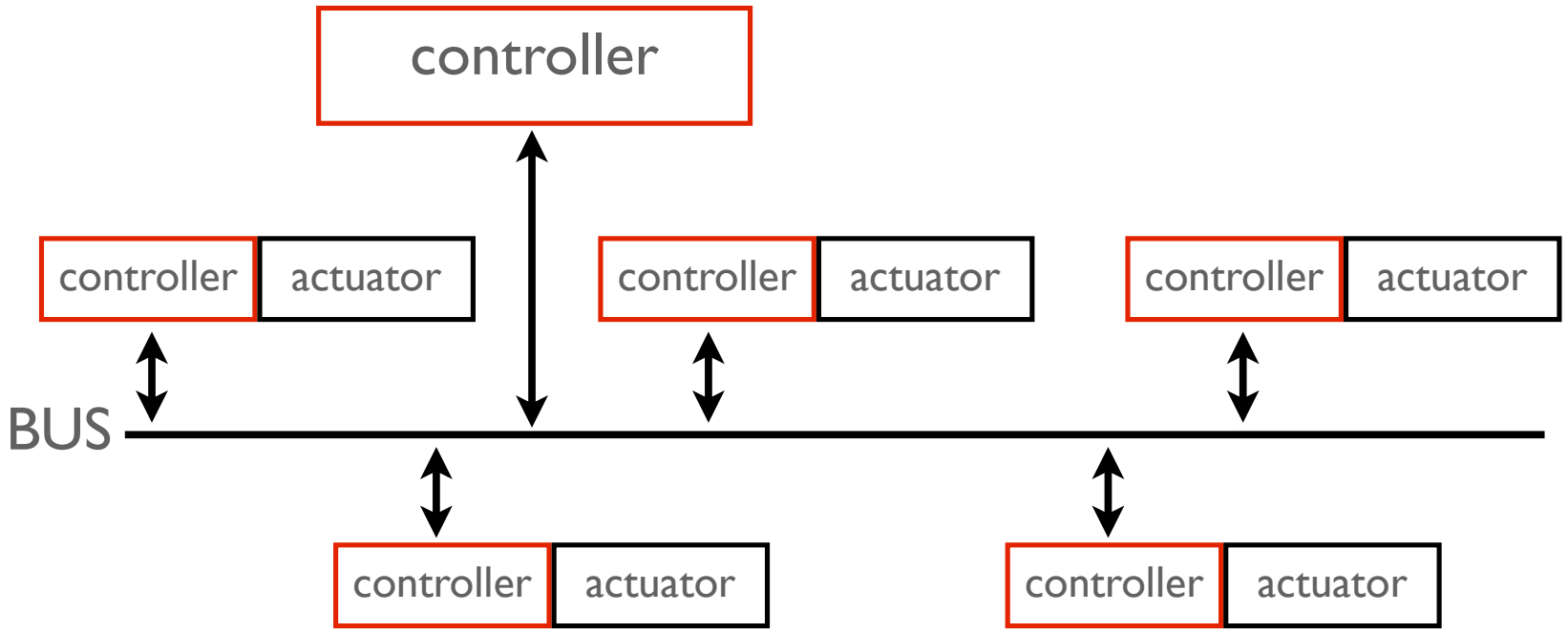
- it can be **distributed** or **decentralised** (or **both?**)

- The control layer: possible architectures

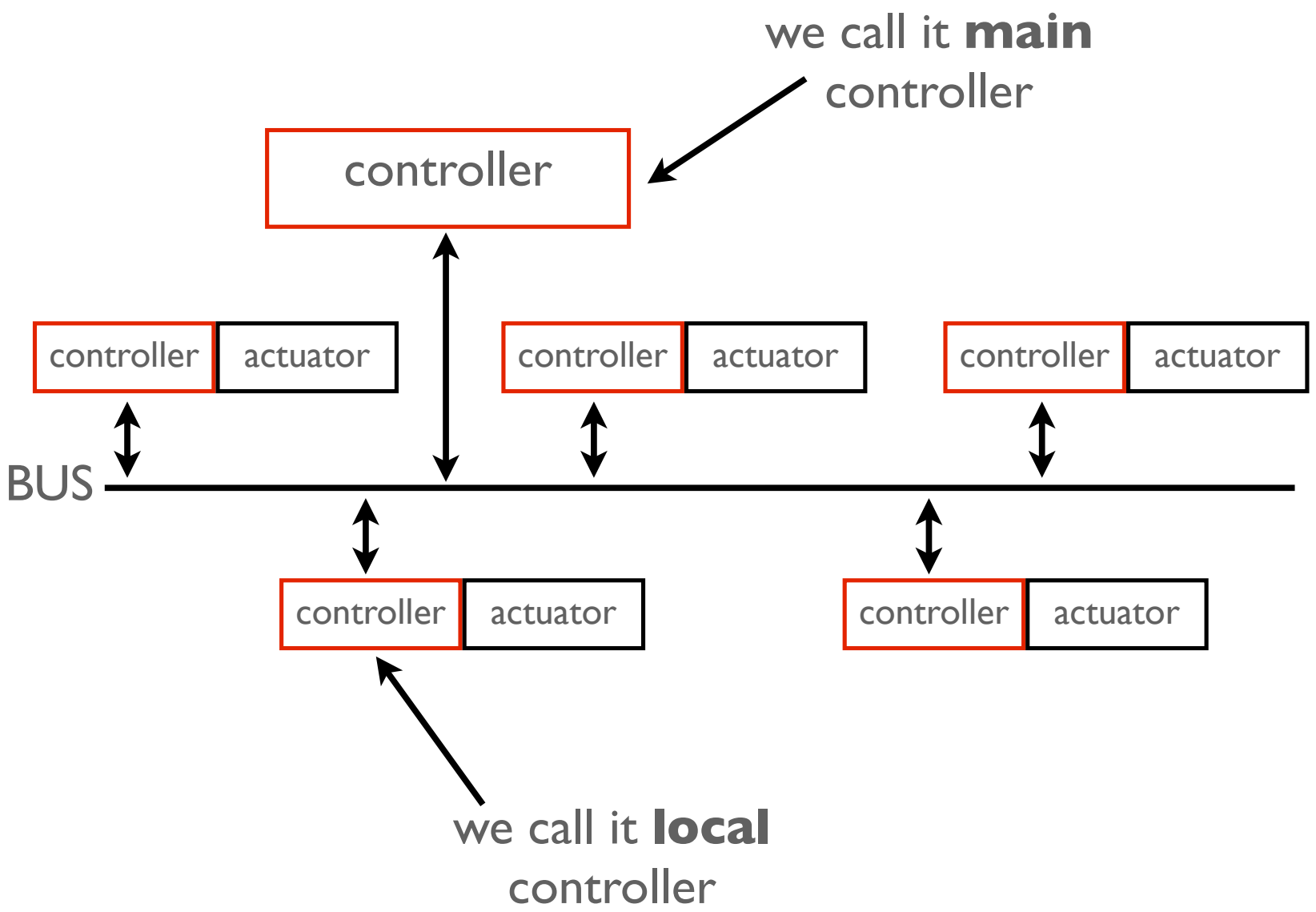




- The control layer: possible architectures

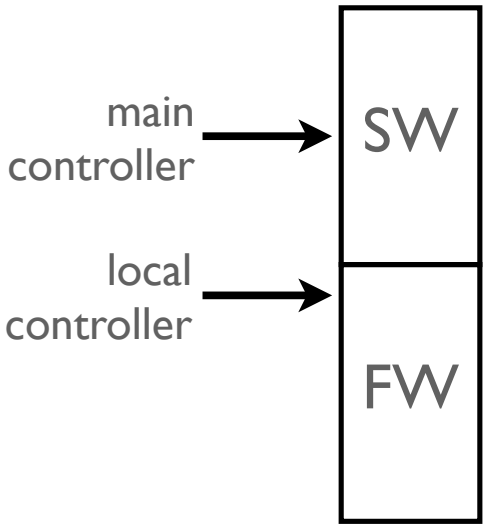


- The control layer: possible architectures



- The control layer: focus on the main controller

  - the main controller is usually a software component
  - the local controller can be firmware or software



  - How to choose the right hardware?
    - an example driven analysis...
    - Alix 3D3
    - Raspberry Pi
    - Jetway NG74

- The control layer: PC Engines Alix 3D3

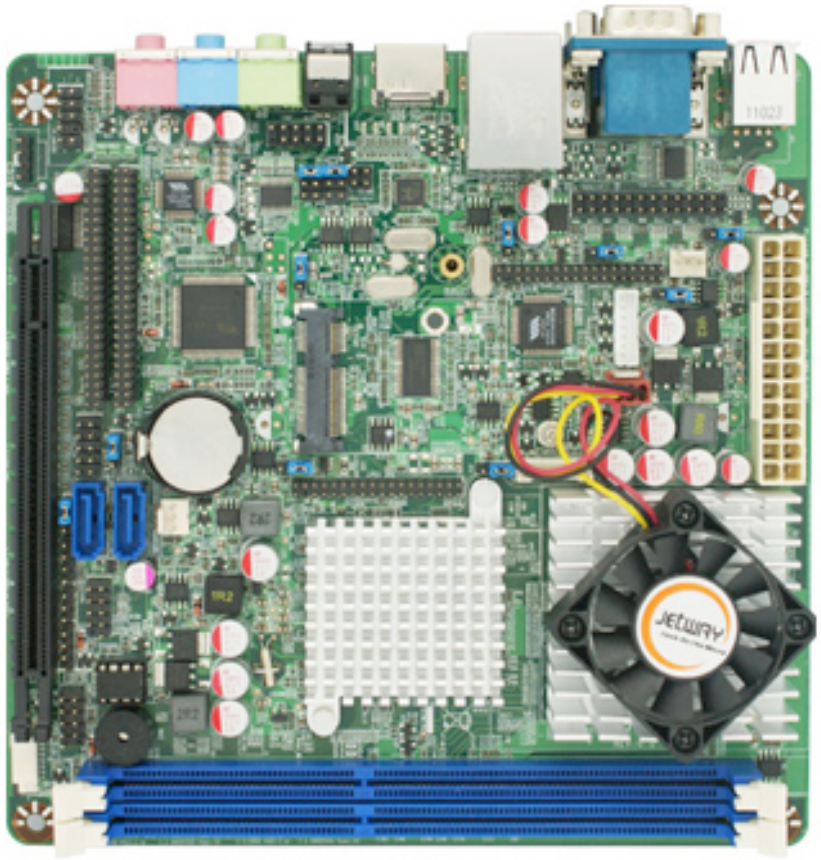
- i86 compatible CPU@500Mhz (AMG Geode)
- 256 Mb RAM
- Flash HDD slot
- fan less
- Connectivity:
  - 1 VGA
  - 2 USB
  - 1 jack audio
  - 1 jack microphone
  - 1 serial
  - 1 ETH
- < 100€ each

- The control layer: Raspberry Pi model B

- ARM CPU@700Mhz (ARM 11 family)
- 256 Mb RAM
- SD HDD slot
- fan less
- Connectivity:
  - 1 mini HDMI
  - 2 USB
  - 1 jack audio
  - 1 ETH
  - 1 GPIO (general purpose I/O)
- < 40€ each

- The control layer: Jetway NG74-2007

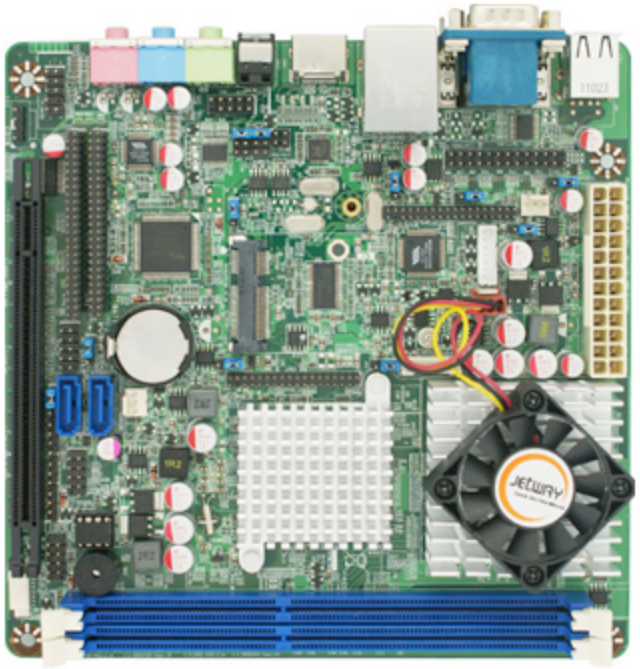
- VIA Nano L2007 CPU@1.6GHz
- up to 8Gb (2 DD3 slots)
- up to TeraBytes (2 serial ATA2)
- Connectivity:
  - 1 HDMI
  - 1 VGA
  - 4 USB
  - 1 ETH
  - 1 serial
  - 3 audio I/O port
  - 1 GPIO
- < 200€ each

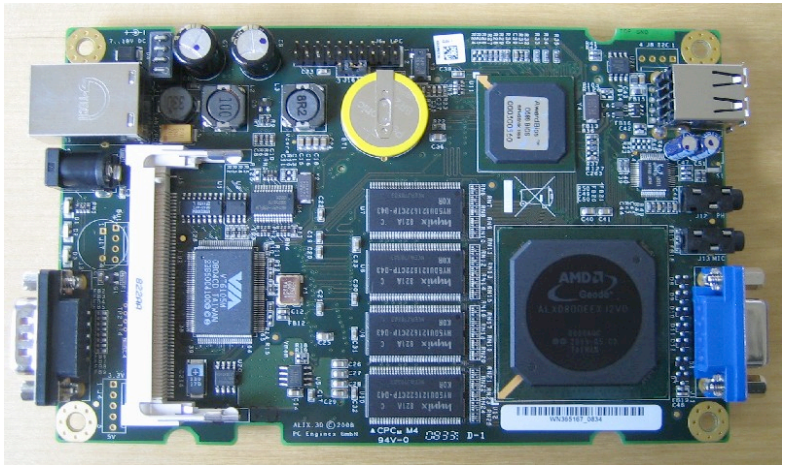


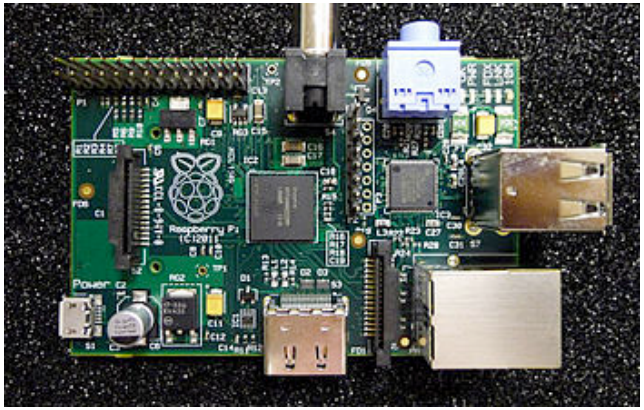
The communication layer: global recap

	Alix	Raspberry	Jetway
CPU	500 MHz	700 MHz	1.6 GHz
RAM	256 Mb	256 Mb	up to 8Gb
HDD	FLASH card	SD card	2 serial ATA
Serial PORT	Yes	NO	Yes
GPIO	NO	Yes	Yes
Cooling	fan less	fan less	fan based
Cost	< 100€	<40€	<200€ (+ optionals)

- The control layer: which is the best Hadware?







(images are not in scale)

- The control layer: how to choose the right software

- the goal is to implement the main controller, the local controllers should be firmware
- the main controller have to:
  - deal with the communication layer
  - deal with the interfaces: both buttons, touch and BCI: a full-OS is required
    - **Linux? Windows embedded? iOs? Android?**
    - iOS and Android have a large resource requirements (because they are designed to deal with graphical interfaces)
    - Linux: reliability++, security-, very light weight version+
    - Windows embedded: reliability-, security+, very light weight version+
    - **Question: why linux seems to be less secure than windows???**

- The control layer: how to choose the right software

- **answer: what about the security of the current kernel in ten years? The kernel of today will still be safe?**
- Anyway linux is the favourite choice...
- And what about the programming language?
  - Assembler, C or Java?
  - Assembler is to complex: it is quite impossible to implement the BCI in assembly
  - C could be the right choice but it is platform-dependent and it is not so easy to integrate with other software
  - Java could also be the right choice, it has a large variety of libraris and softwares ready-to-use but it has worse performance than C
- What-if we decide for java and then we move to C?
  (or the opposite...) decoupling, decoupling, decoupling...

- The control layer: proxy and semantic repository: how to decouple the control layer from the actuation layer

- ...Mario Caruso...

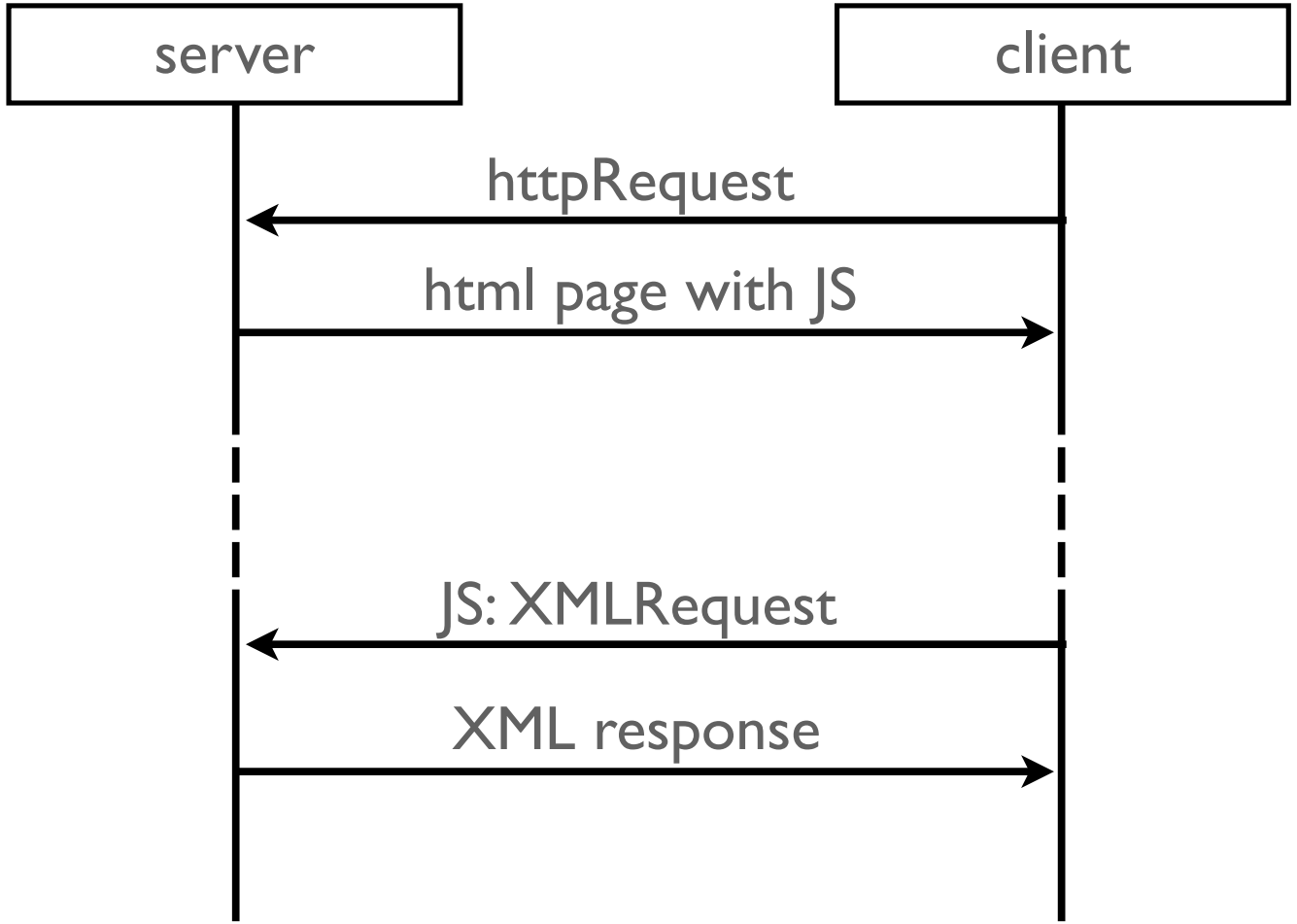
- The control interface layer: how to implement the push notification in the browser

- the web is client-server oriented, ***a browser will never accept connections if it does not start the communication***...

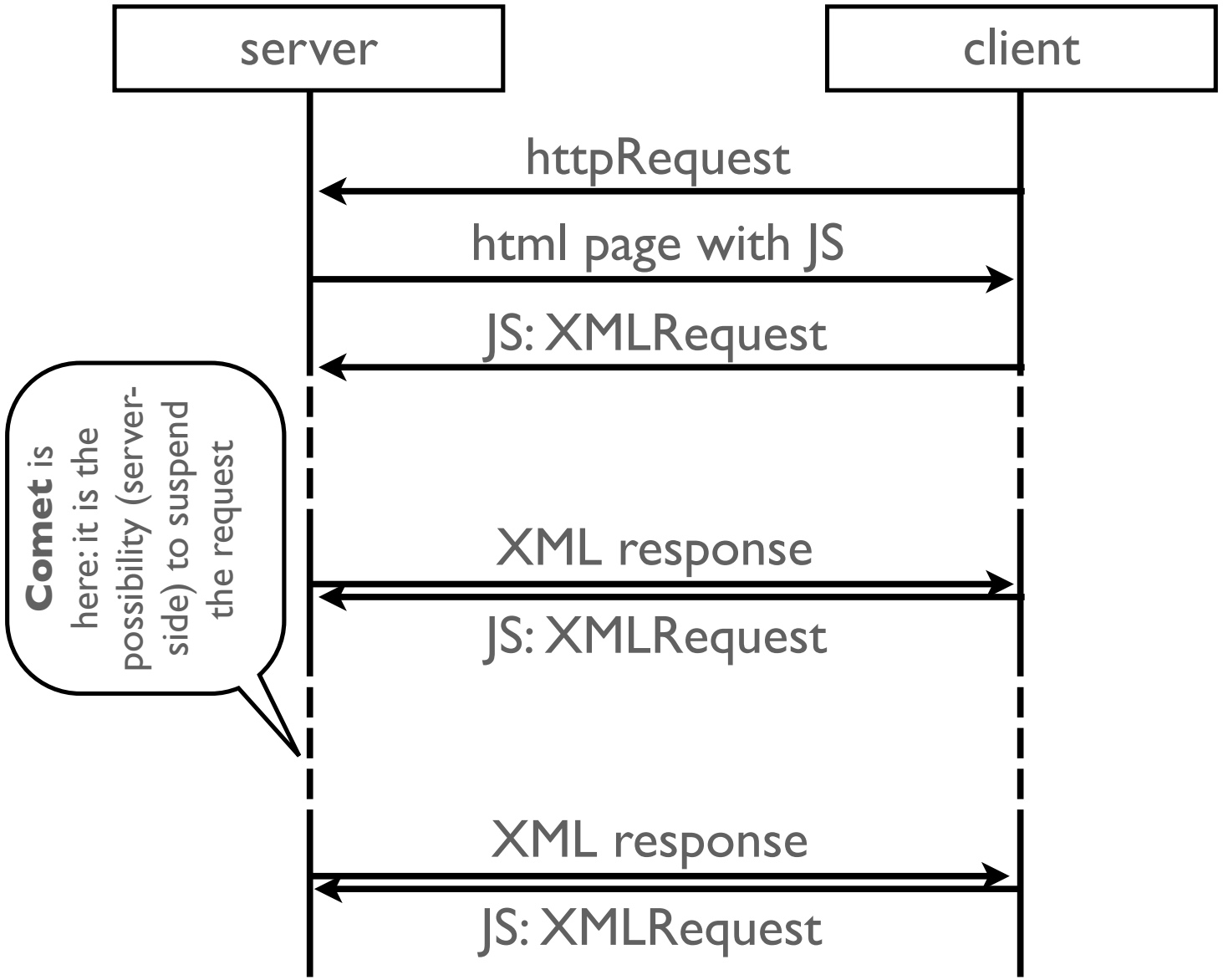
- so how can we implement a push notification in the browser?
  - a work-around is required...

- push notifications become possible thanks to:
  - Ajax, i.e. XML request from javascript
  - Comet (also known as reverse Ajax)

- The control interface layer: Ajax



- The control interface layer: from Ajax to Comet



The control interface layer: from Ajax to Comet

Server Side:

```
@Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        Continuation cc = ContinuationSupport.getContinuation(req, null);
        CometSessionObject r = (CometSessionObject) cc.getObject();
        if (r == null) {
            CometSessionObject appo = new CometSessionObject();
            cc.setObject(appo);
        }
        add(cc);
        cc.suspend(TIMEOUT);
        r = (CometSessionObject) cc.getObject();
        resp.getWriter().write(r.getModule() + "-" + r.getState());
        resp.getWriter().flush();
        resp.getWriter().close();
    }
```

- The control interface layer: from Ajax to Comet

- Server Side:

```
private void add (Continuation c) {
        this. continuationSet.add(c);
}

public void uponEvent(event e) {
     itc = continuationSet.iterator();
     while (itc.hasNext()) {
          Continuation c = itc.next();
          CometSessionObject appo = (CometSessionObject) c.getObject();
          appo.setModule(e.getName());
          appo.setState(e.getValue());
          c.setObject(appo);
          c.resume();
     }
}
```

- The control interface layer: from Ajax to Comet

- Client Side:

```
function startPushNotification() {
    handlerRA = new XMLHttpRequest();
    handlerRA.open("GET","/comet/pushNotification", true);
    handlerRA.onreadystatechange = pushNotification;
    handlerRA.send();
}

function pushNotification() {
    if (handlerRA.readyState == 4) {
        var rText = handlerRA.responseText;
        handlerRA = new XMLHttpRequest();
        handlerRA.open("GET","/comet/pushNotification", true);
        handlerRA.onreadystatechange = pushNotification;
        handlerRA.send();
        //rText contains the push notification
    }
}
```