LCM

The Network Is NOT Transparent

Maurizio Martignano
Spazio IT – Soluzioni Informatiche s.a.s
Via Manzoni 40
46030 San Giorgio di Mantova, Mantova
http://www.spazioit.com

October 2015

Agenda

- CRETA Platform
- LCM
- LCM and FDIR
- Conclusions

CRETA Platform

CRETA Platform – What is it?

- Avionics architectures can contain nowadays a variety of communication hardware, e.g.:
  - AFDX (Avionics Full-Duplex Switched) Ethernet
  - ARINC 429, MIL-STD-1553B,…
  - RS 232, RS 422, RS485,…
- In turn, different types of computers, controllers, devices etc…, are connected to the communication infrastructure.
- Finally various types of software/firmware systems run on the adopted computers and controllers. These systems are not necessarily written in the same programming language.

CRETA Platform – What is it?

- When developing and/or maintaining Software Components belonging to a given architecture it is very important to be able to integrate them with each others and to verify how well they interoperate together.
- And the sooner this verification activity is performed, the better.

- CRETA (**Cr**oss **E**nvironment for **T**est and **A**nalysis) is a software platform simulating both:
  - the avionics communication infrastructure
  - the various avionics computational environments

CRETA Platform – What is it?

- With CRETA software developers/maintainers can «check» their software component directly on their workstation, on a simulated «testbed».
- Tests and verifications on the actual hardware «testbed» can be performed at later time, once the software component has successfully proved its interoperability on CRETA.
- Actual hardware can also be integrated to CRETA, as soon as it available (or the other way around: CRETA can be integrated to actual hardware).

CRETA Platform – Programmatics

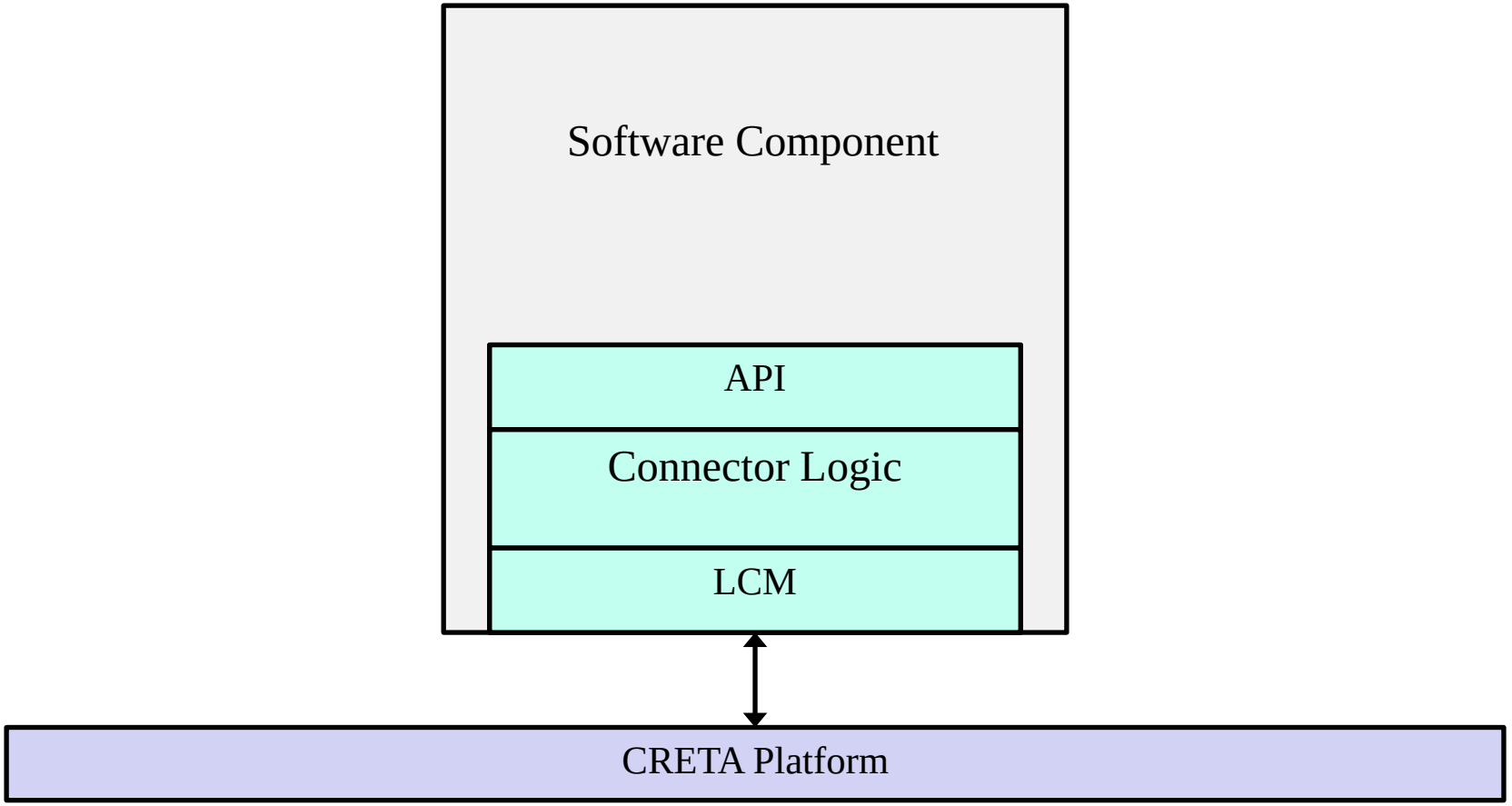
- CRETA is project under development funded by AIRBUS Helicopters.

- The main CRETA Developers are:
  - Inopus (http://www.inopus.de/)
  - Spazio IT (http://www.spazioit.com)

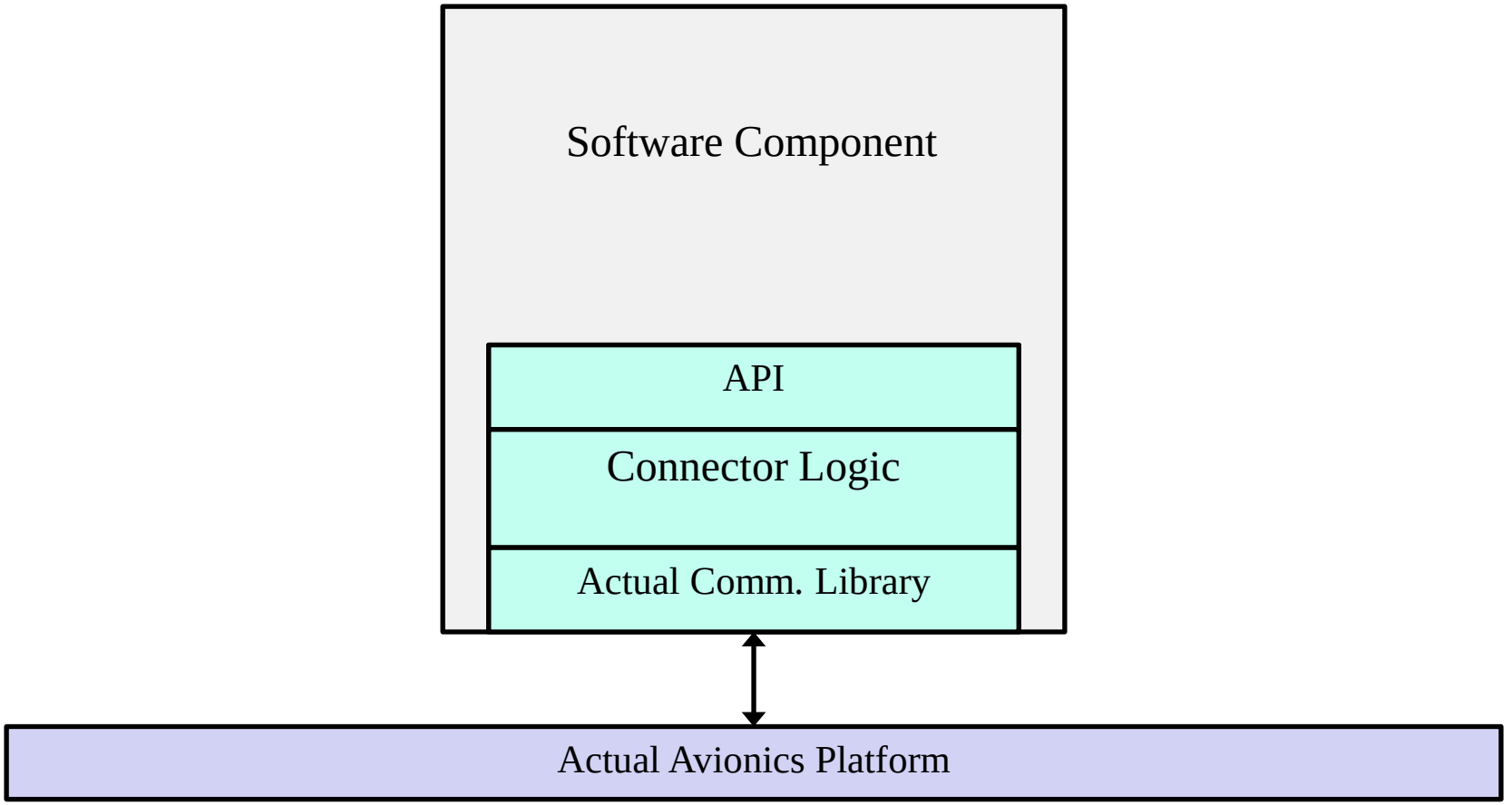
CRETA Platform – LCM

- Software components «attached» to CRETA can run on different workstations in a networked environment.
- CRETA exposes itself to the software component via a «connector», i.e. a standard API.
- The «connector» in turn connects to the CRETA system and other software components via an internal middleware based on LCM.

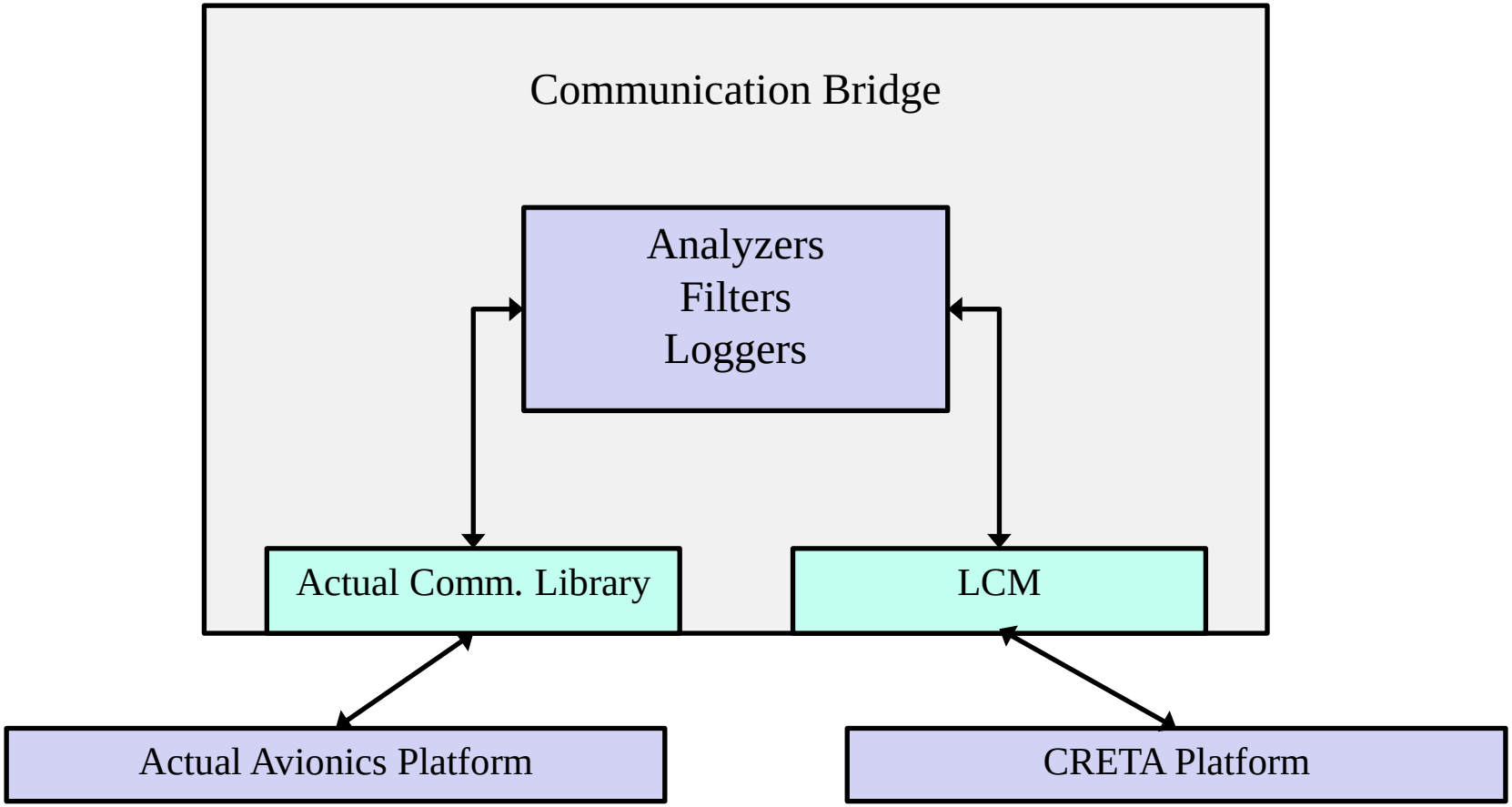
CRETA Platform – LCM

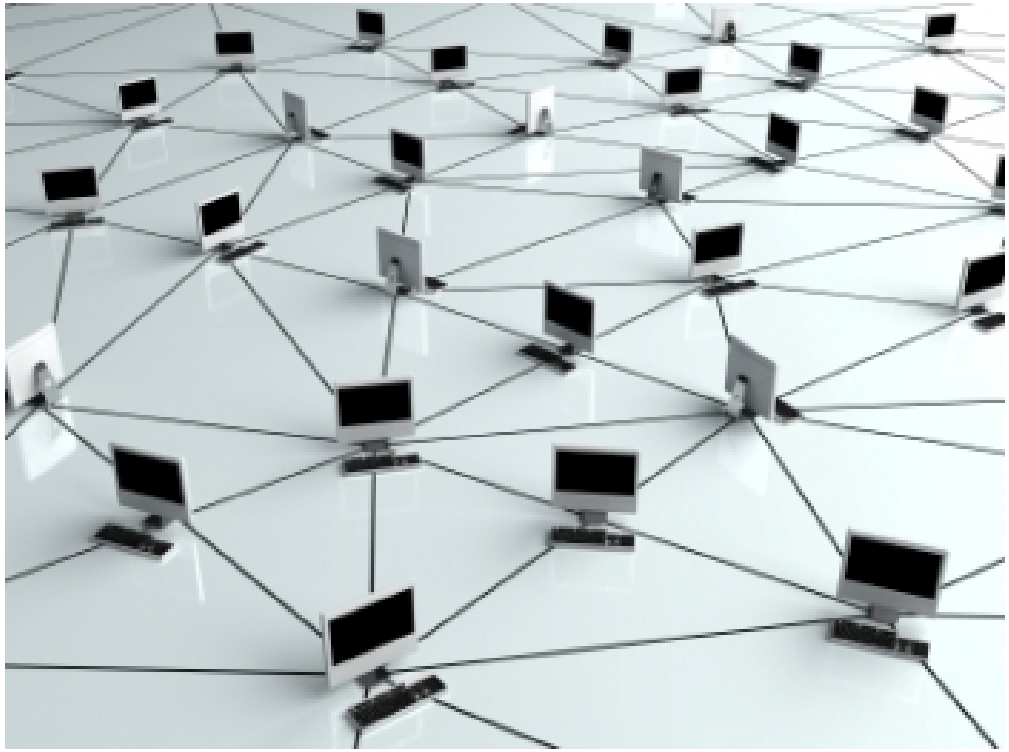


CRETA Platform – LCM



CRETA Platform – LCM





Lightweight Communications and Marshalling

LCM

- “LCM was created for use on MIT's DARPA Urban Challenge vehicle, with development starting during the summer of 2006.”

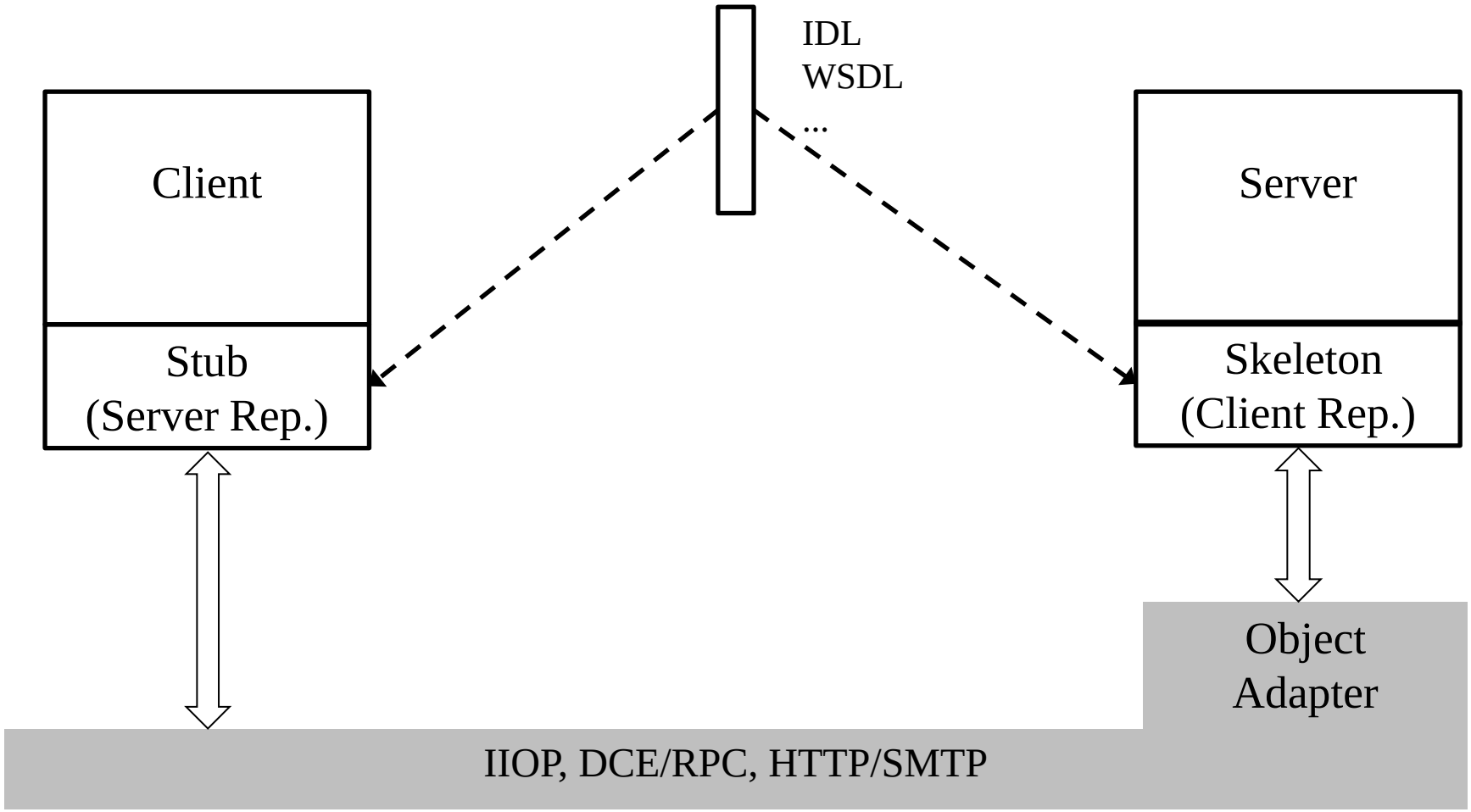
- LCM is publically available at: https://lcm-proj.github.io/

- LCM supported languages are: C, C++, C#, Java, Lua, MATLAB, Python

LCM – Design Goals

- Communication
  - Supports many-to-many communication.
  - No per-network or per-host daemon needed for relaying data
  - Each packet transmitted appears on the wire no more than one time
  - Minimal latency
  - Scales to high bandwidth
- Marshalling
  - Absolute type-safety
  - Platform- and language-independence
  - Ease of use
- FDIR
  - Does not guarantee delivery of a particular message.
  - Does not guarantee ordering of messages.

RPC
(the network is transparent)



RPC
(the network is transparent)

- Local call

```
…
local_foo();
…
```

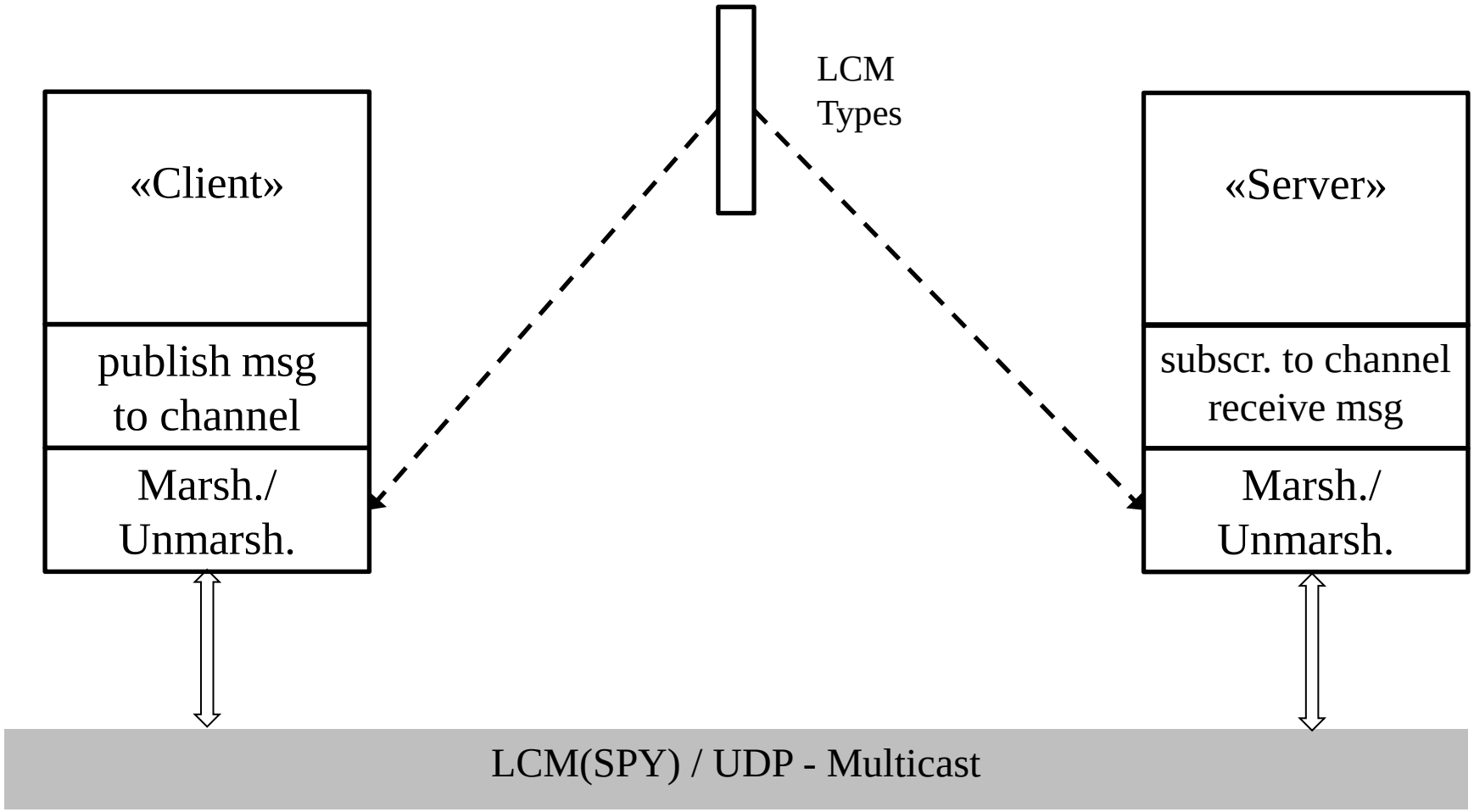
- Remote call

```
…
try {
  remote_foo();
} catch (Local_Remote
Exception ex) {
  // Something here
 // … (but what?)
}
```

RPC
(the network is transparent)

- Characteristics
  - Local / Remote Exceptions
  - Indirect Memory Allocation
  - Blocking Calls
  - (Stub / Skeleton Binding at Compile Time)

LCM
(the network is NOT transparent)



LCM
(the network is NOT transparent)

- Local call

```
…
local_foo();
…
```

- Message publishing

```
…
example_t msg = new
example_t();
// … initialize message
try {
  lcm.publish("EXAMPLE",
msg);
 } catch (LocalException ex) {
// Something here
}
```

LCM
(the network is NOT transparent)

- Characteristics
  - Local Exceptions
  - No Indirect Memory Allocation
  - No Blocking Calls
  - (Message Types Binding at Compile Time)

LCM and FDIR

LCM and FDIR

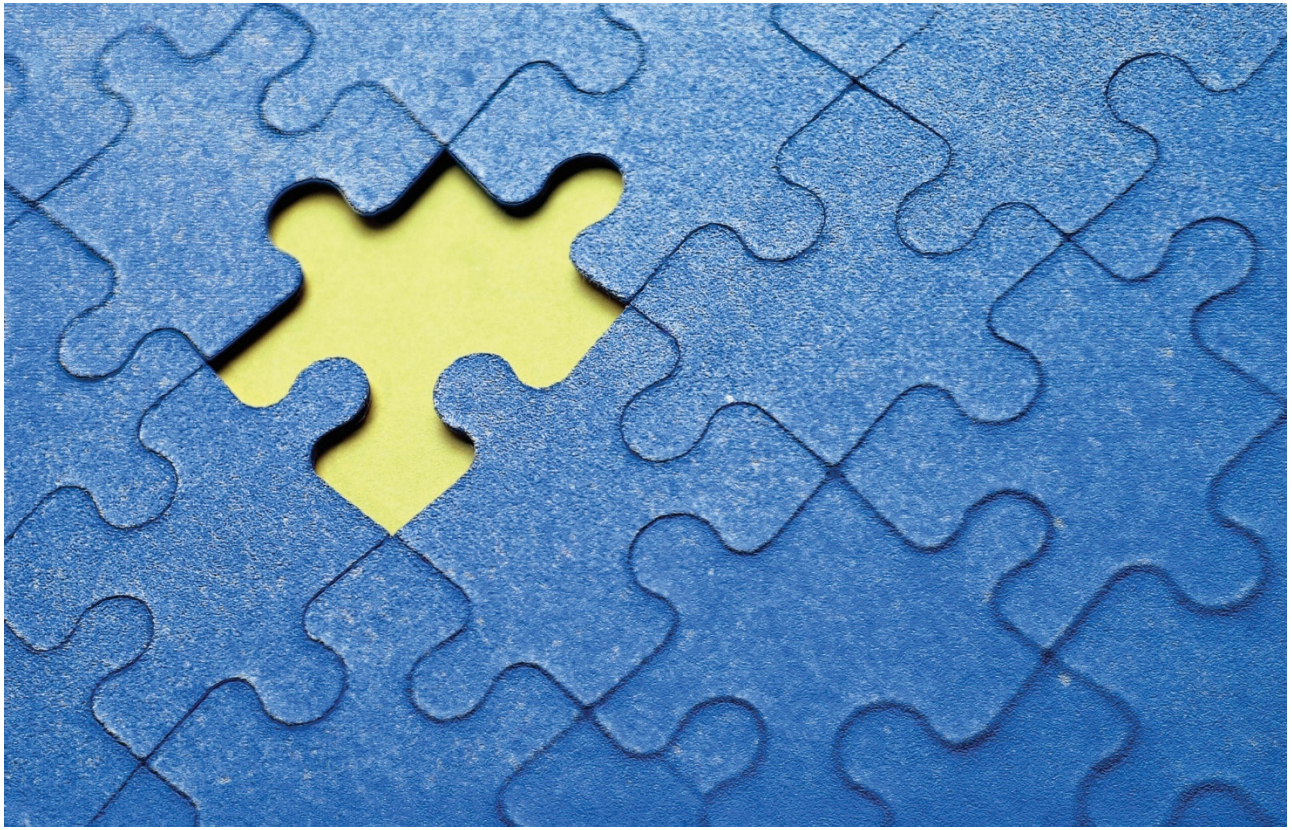
- LCM does not rely on «servers»:
  - Supports many-to-many communication.
  - No per-network or per-host daemon needed for relaying data
- LCM does not guarantee the communication has taken place:
  - Does not guarantee delivery of a particular message.
  - Does not guarantee ordering of messages.

LCM and FDIR

- Additionally, “LCM provides several tools useful for logging, replaying, and inspecting traffic.

- These tools allow developers to quickly identify many of the most common failures. “ and can be used to implement failure detection mechanisms.

Conclusions



Conclusions

- Complex software systems can be decomposed in a set of interconnected processes.
- The fault isolation provided by a set of processes connected by a proper communication infrastructure can be comparable to the one offered by virtual machines and/or containers.
- Not relying on central/critical components as well as non relying on particular assumptions / guarantees increases the overall robustness of the system.
- The network is not transparent.

References

- http://people.csail.mit.edu/albert/pubs/2010-huang-olson-moore-lcm-iros.pdf
- http://lcm.googlecode.com/svn-history/r452/www/reference/lcm/intro.html

Thank you for your attention