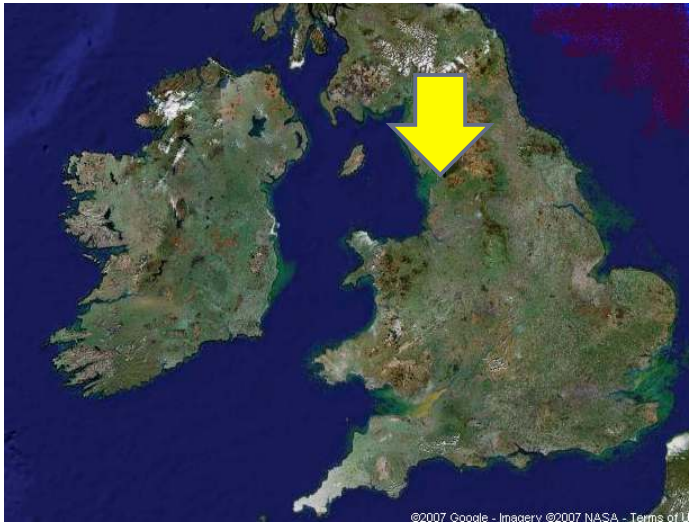
Deep & wide
Distributed programming in a large-scale fast-moving world

François Taïani

InfoLab21
Computing Department

Introducing Lancaster



Lancaster Middleware Group

What we do …

- Facilitate development of **new distributed systems**
  - ➔ Abstractions?
  - ➔ Mechanisms?
  - ➔ Exotic environments: MANETs, WSN, routers, …

Next-gen middleware team

- **Academics**
  - Gordon Blair
  - Geoff Coulson
  - Francois Taiani
- **Researchers**
  - Paul Grace
  - Barry Porter
  - Nelly Bencomo
- **Students**
  - Shen Lin , Lei Liu, Vatsala Nundloll, Rajiv Ramdhany, Bholanath Surajbali, Nirmal Weerasinghe …

Barry

InfoLab21
Computing Department

Current research targets

- A **wide** computing world:
  - broad, pervasive, and distributed
  - e.g. MANETs, grids, clouds, VLN, social networks
  - Which **abstractions** for a **multifaceted** reality?
- A **deep** computing world:
  - drilling into the small: WSN, ECU, microchips, VMM
  - transformative jumps at any level
  - Which **abstraction** for a **multi-layer** reality?
- **Large** scale and **fast** moving
  - Issue for machines (physical) and **humans (logical)**

Outline

Helping to realise distributed systems in:

- a **wide** world
- a **deep** world
- built (for and) **by humans**

Outline

Helping to realise distributed systems in:

- a **wide** world
- a **deep** world
- built (for and) **by humans**

A wide world

Co-existing technologies, systems, & applications

- Should work together: **interoperability**
- Should thrive together: **synergies**

Interoperability

- **Multi-faceted** 'Sys. of Systems'
  - → together to meet global aims
- Extreme **heterogeneity**
  - → Techno, network, sw, hw
- **Dynamic** composition
  - → unknown until runtime

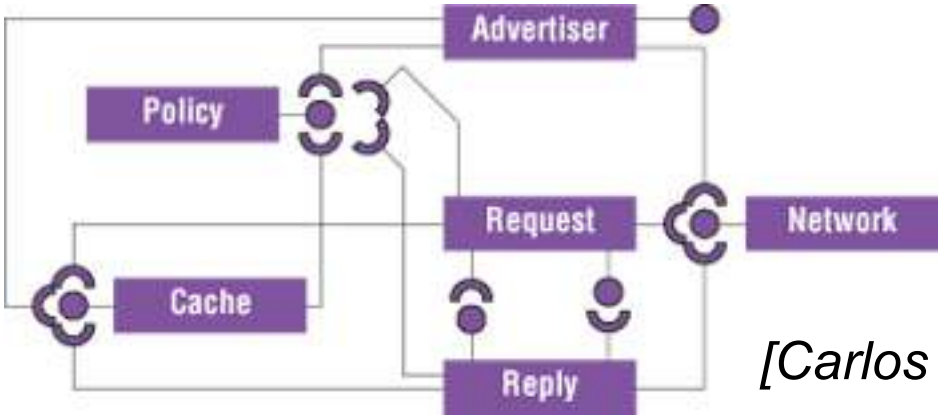
Vision: On-the-fly Interop.

→ **Just-in-time middleware**

Example: Service Discovery Protocols

Interoperable Service Discovery

- Lancaster approach: **deconstruction reconstruction**
  - ➔ **decompose** set of SD protocols (SDP, UPnP, etc.)
  - ➔ **generic** component based **architecture**

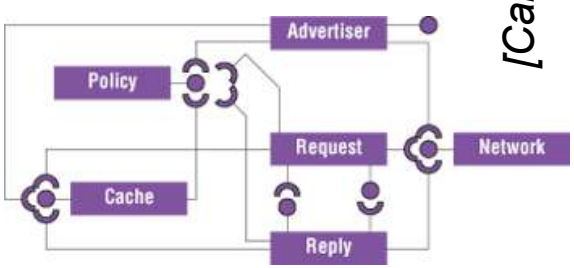


[Carlos Flores 07]

Interoperable Service Discovery

- Lancaster approach: **deconstruction reconstruction**
  - ➔ **decompose** set of SD protocols (SDP, UPnP, etc.)
  - ➔ **generic** component based **architecture**
  - ➔ specific front-facing **instances** for each protocol
  - ➔ specific configuration for each **roles**

[Carlos Flores 07]



- **Next step:** for any kind of middleware paradigm
  - ➔ on-the-fly synthesis of connectors (from protocol spec.)
  - ➔ **Connect Project** (FP7), ongoing

A wide world

Co-existing systems, technologies, & applications

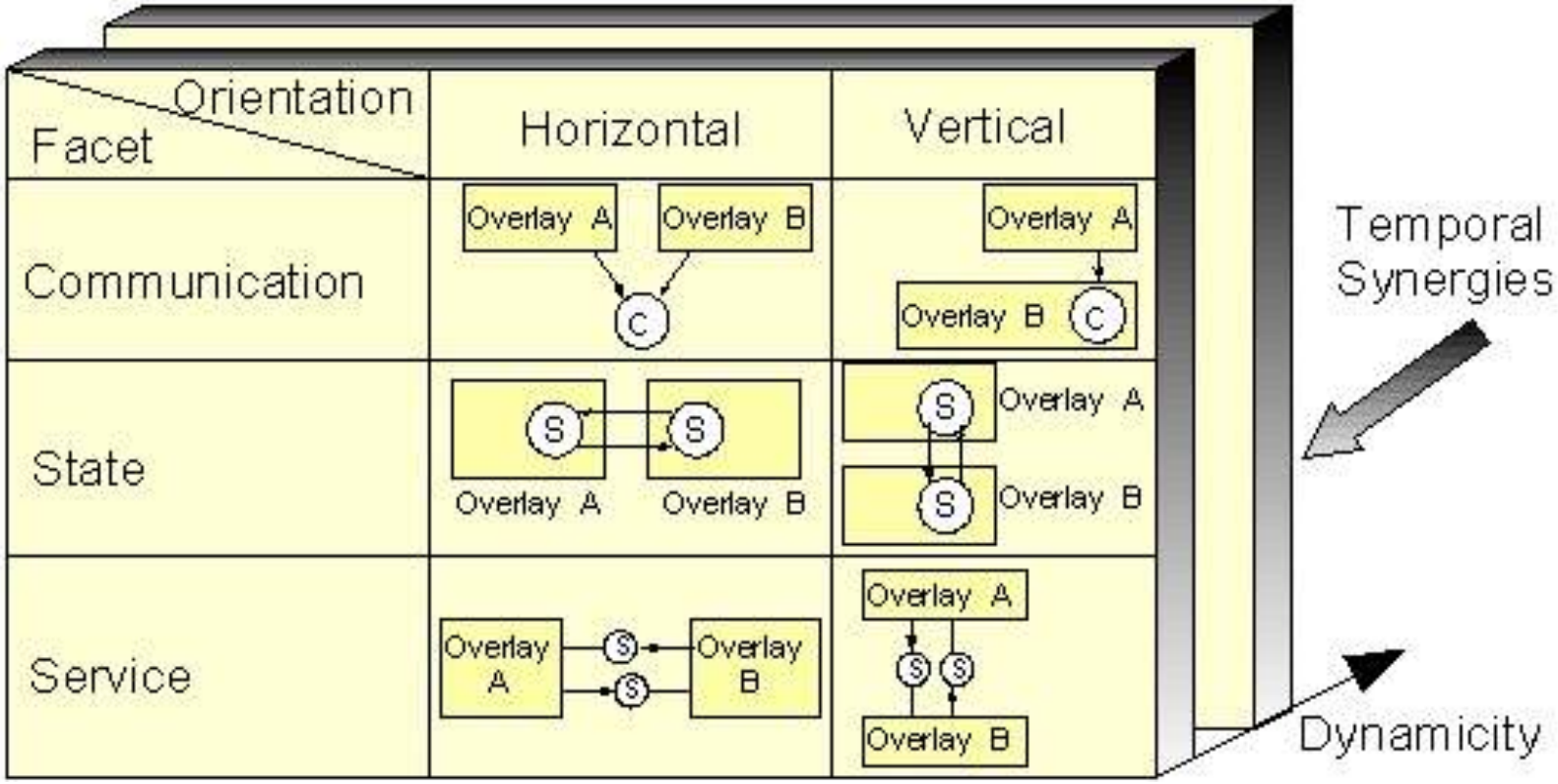
- Should work together: **interoperability**
- Should thrive together: **synergies**

Synergies

Goal: get independent distributed systems to **cooperate**

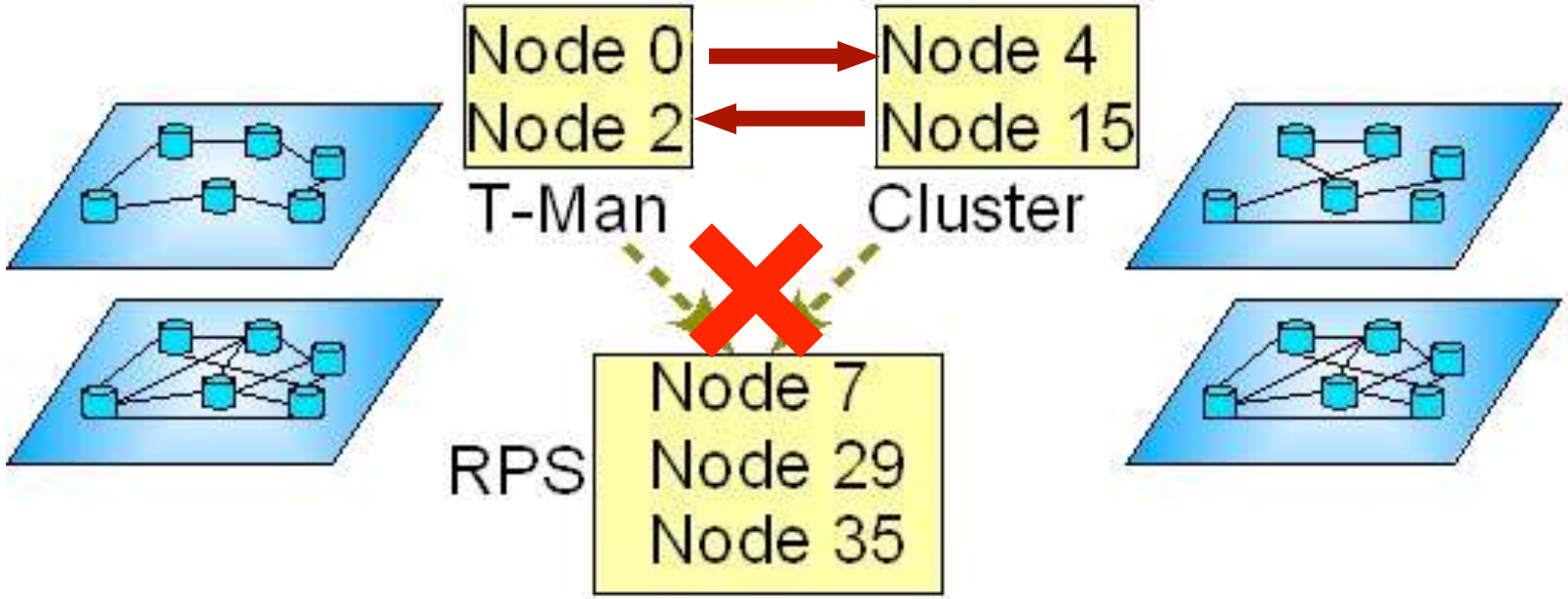
- **Synergy = Local** Behaviour, **Global** benefits
  - ➔ Local adaptation of each overlay's behaviour
  - ➔ Goal: Improve overall efficiency
  - ➔ High payoff (global) at a minimum cost (local)
- **Systematic analysis** of synergies in overlays
  - ➔ assuming component-based architecture
  - ➔ both in space *and* time
- **Challenge**
  - ➔ conserve original properties of combined systems

A Classification of Synergies



[Shen Lin 09]

Horizontal State Synergy

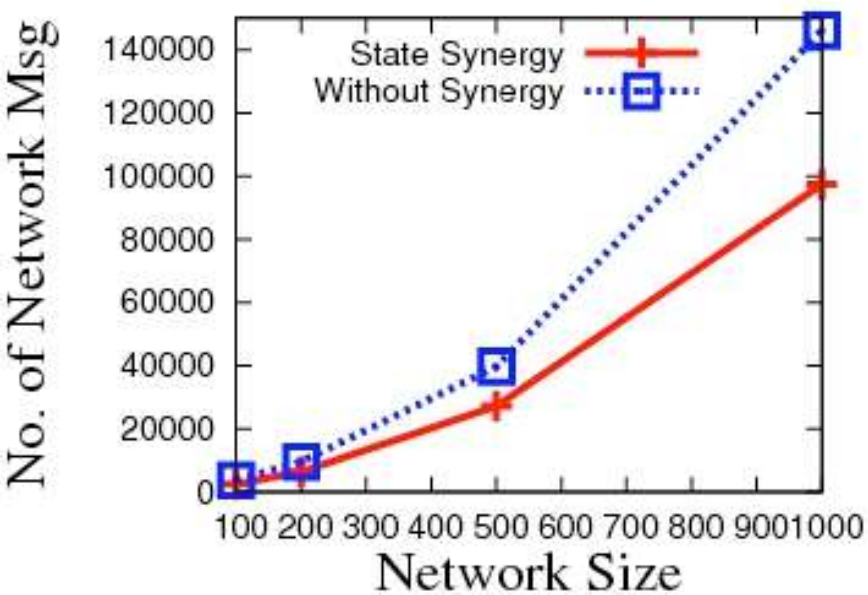


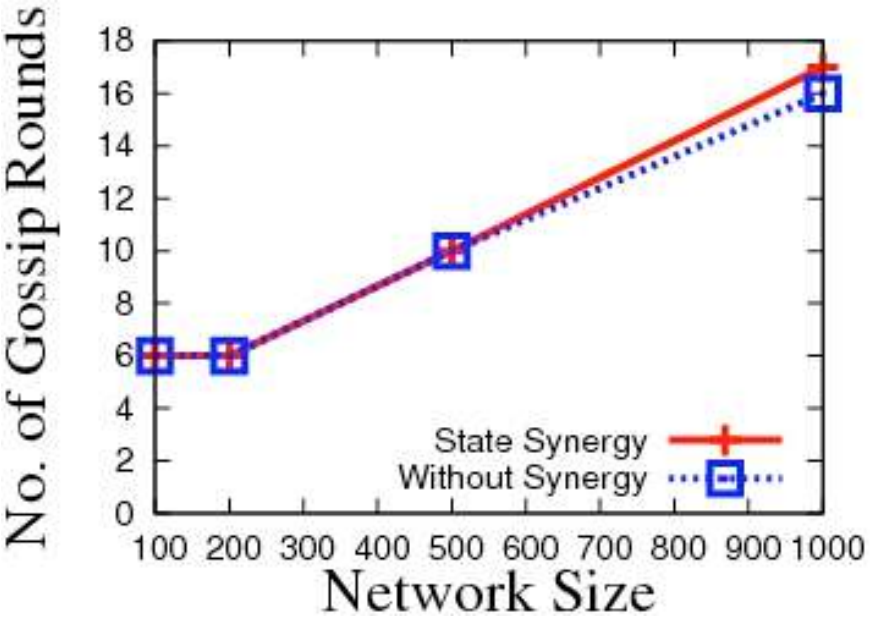
- **Potential Benefit:** message reduction by 1/3
- **Potential Risk:** failure to converge

Horizontal State Synergy

Simulation on Jist/SWANS (Cornel)

GossipKit Component framework





- Messages reduced by 35%
- Convergence maintained

Outline

Helping to realise distributed systems in:

- a **wide** world
- a **deep** world
- built (for and) **by humans**

The Wisebed approach (FP7)

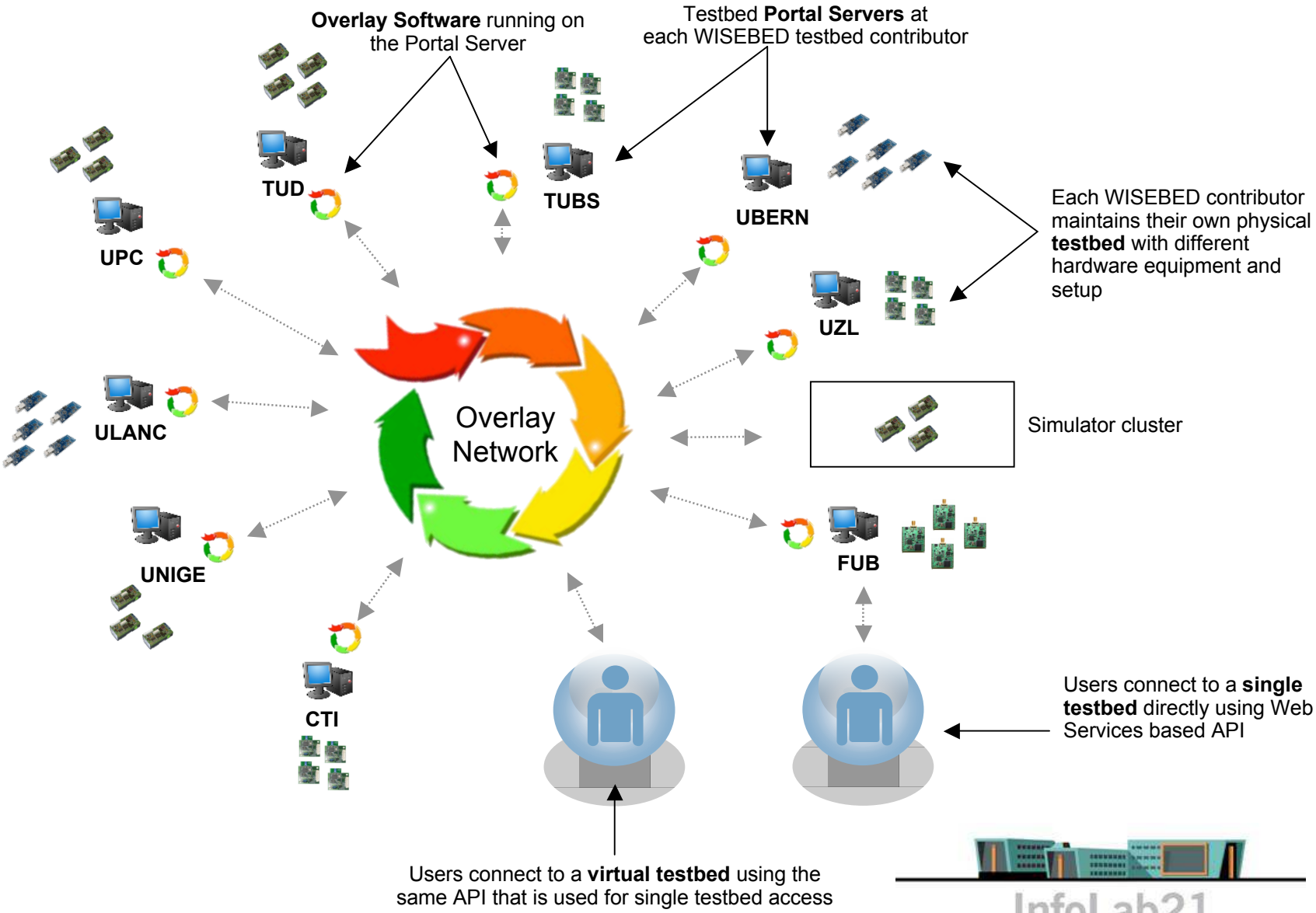
Goal: large-scale WSN test beds

- configurable, extensible, federated

Approach

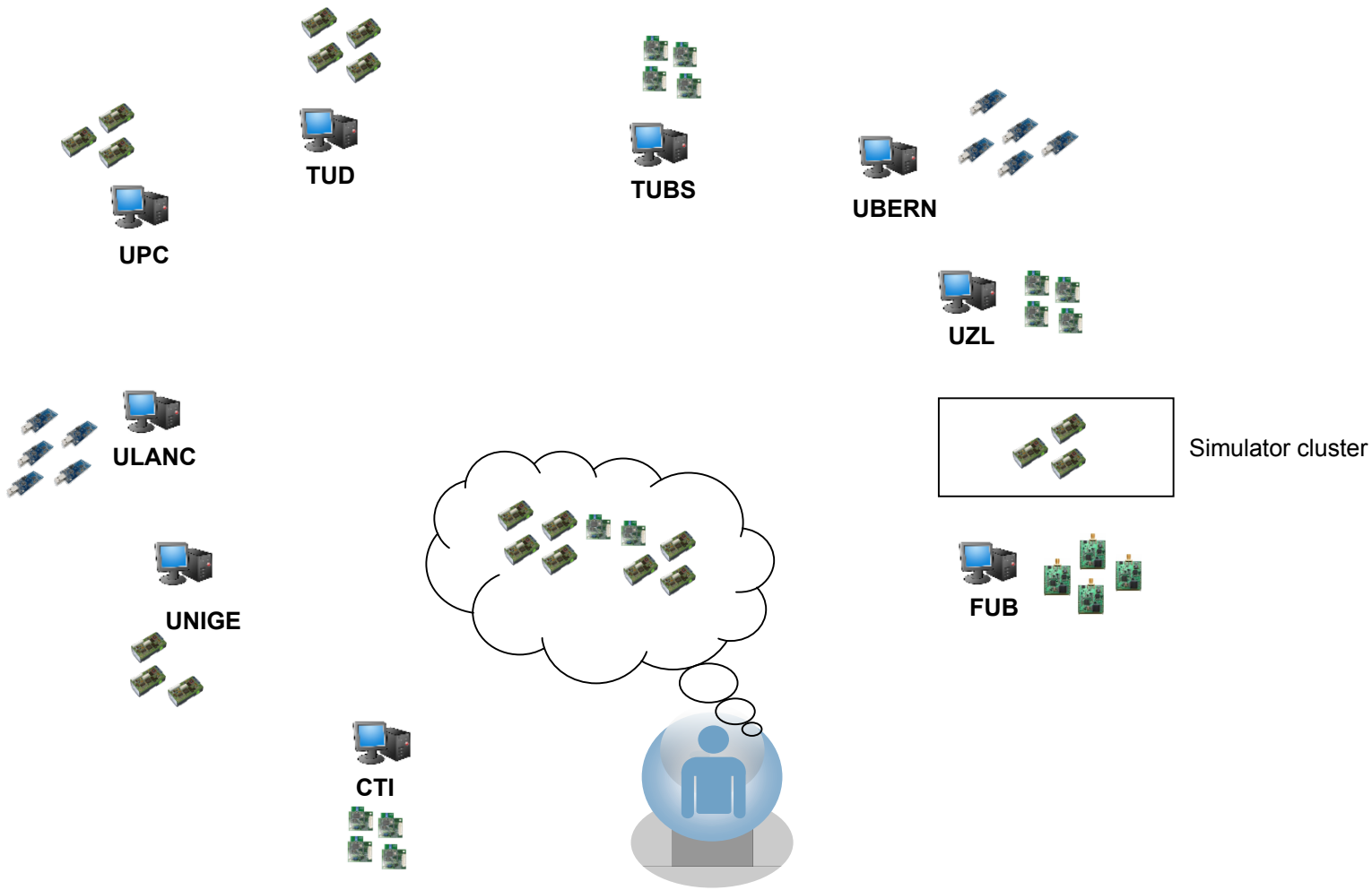
- **Interconnect** individual testbeds via the Internet
- **Mix** physicality, virtuality and simulation
- Offer value-added **software services**
  - ➔ A library of algorithms, mechanisms and protocols
  - ➔ A SDK tailored for heterogeneity and change **[Lorien]**
- Support **management** of data traces

The structure of Wisebed



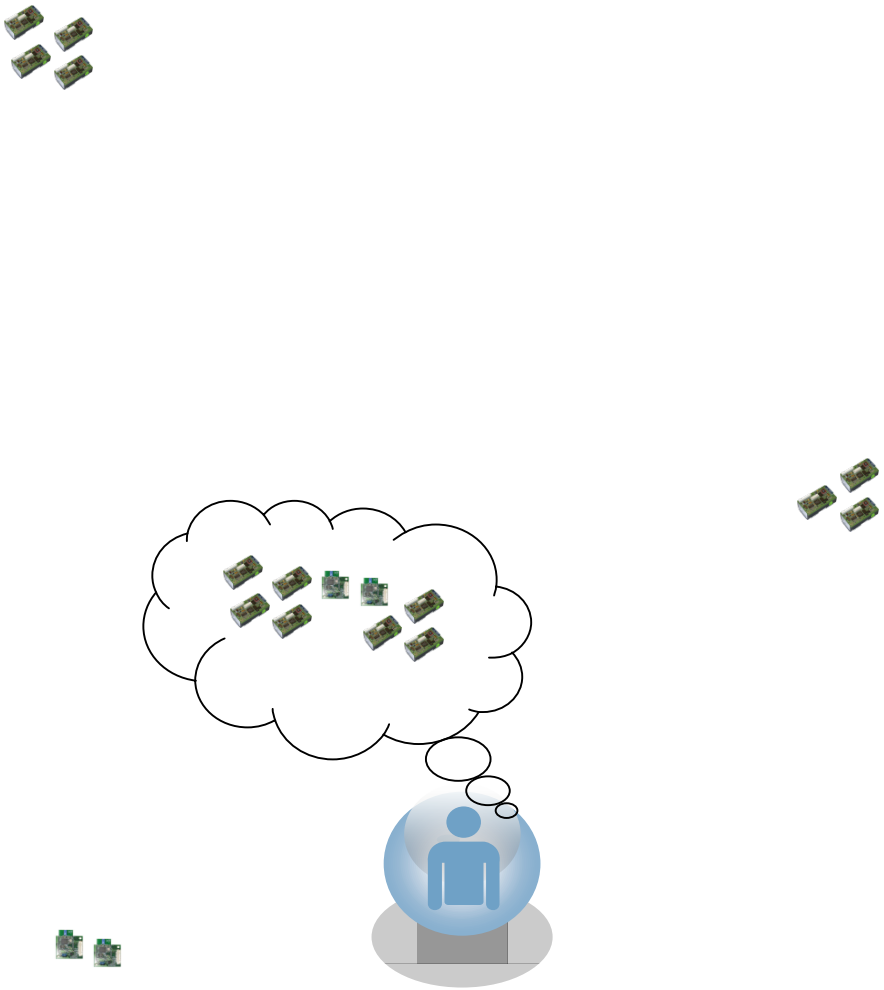
InfoLab21
Computing Department

The structure of Wisebed



InfoLab21
Computing Department

The structure of Wisebed



Hard and soft heterogeneity

Different CPUs

Different Languages

Different amount of RAM

Platform	CPU	OS	Language	D	RAM	Bits
iSense	Jennic	iSense-FW	C++	128kB	92kB	32
ScatterWeb MSB	MSP430	SCW-FW		48kB	10kB	16
ScatterWeb ESB	MSP430	SCW-F	None	60kB	2kB	16
Tmote Sky	MSP430		C	Physical	48kB	10kB	16
MicaZ	ATMega128		C	Physical	128kB	4kB	8
TNode	AT	yOS	nesC	Physical	128kB	4kB	8
iMote2		TinyOS	nesC	Physical	32MB	32MB	32
GumStix	Scale	Emb. Linux	C	Virtual	16MB	64MB	32
Shawn	Desktop	Win/Linux/Mac	C++	Virtual	∞	∞	32/64
ns-2	Desktop	Win/Linux/Mac	C++/OTcl	Virtual	∞	∞	32/64

Highly flexible SDK needed

Different OSs

Different amount of ROM

A WSN-OS: Lorien

- Lorien's is (part of) Wisebed's WSN-SDK
  - → **pure component-based** OS
  - → each part strongly-separated / dynamically replaceable

- Built using **OpenCom** (C version)
  - → **minimalist / purist** approach
  - → OpenCom kernel is a component like any other
  - → **OpenCom is OS-less**: runs on bare metal
  - → 'proto-kernel'

[Barry Porter 09]

Concrete realisation on TelosB

- GSC, configuration files & roles

[48KB program memory, 10KB RAM]

SCF.cfg 0

[Roles]
ROMFS=linfs.so
MassStorage=lexfs.so
Loader=loader.so
Runtime=opencom.so
GSC=gsc.so

[Bindings]
Loader.IIOFile->MassStorage
Loader.IIOBlockNamed->ROMFS
Runtime.IElfLoader->Loader
GSC.IIOBlockNamed->ROMFS
GSC.IIOFile->MassStorage
GSC.IElfLoader->Loader

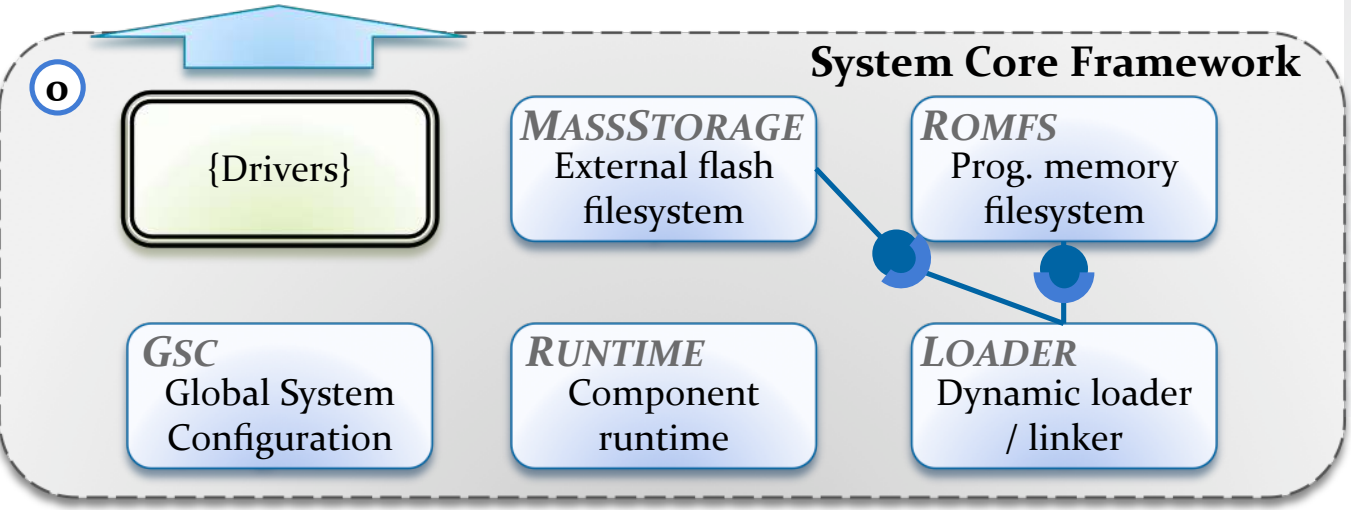
[Frameworks]
Drivers.cfg 1

GSC
Global System Configuration

Concrete realisation on TelosB

- GSC, configuration files & roles

[48KB program memory, 10KB RAM]



SCF.cfg 0

[Roles]

ROMFS=linfs.so
MassStorage=lexfs.so
Loader=loader.so
Runtime=opencom.so
GSC=gsc.so

[Bindings]

Loader.IIOFile->MassStorage
Loader.IIOBlockNamed->ROMFS
Runtime.IElfLoader->Loader
GSC.IIOBlockNamed->ROMFS
GSC.IIOFile->MassStorage
GSC.IElfLoader->Loader

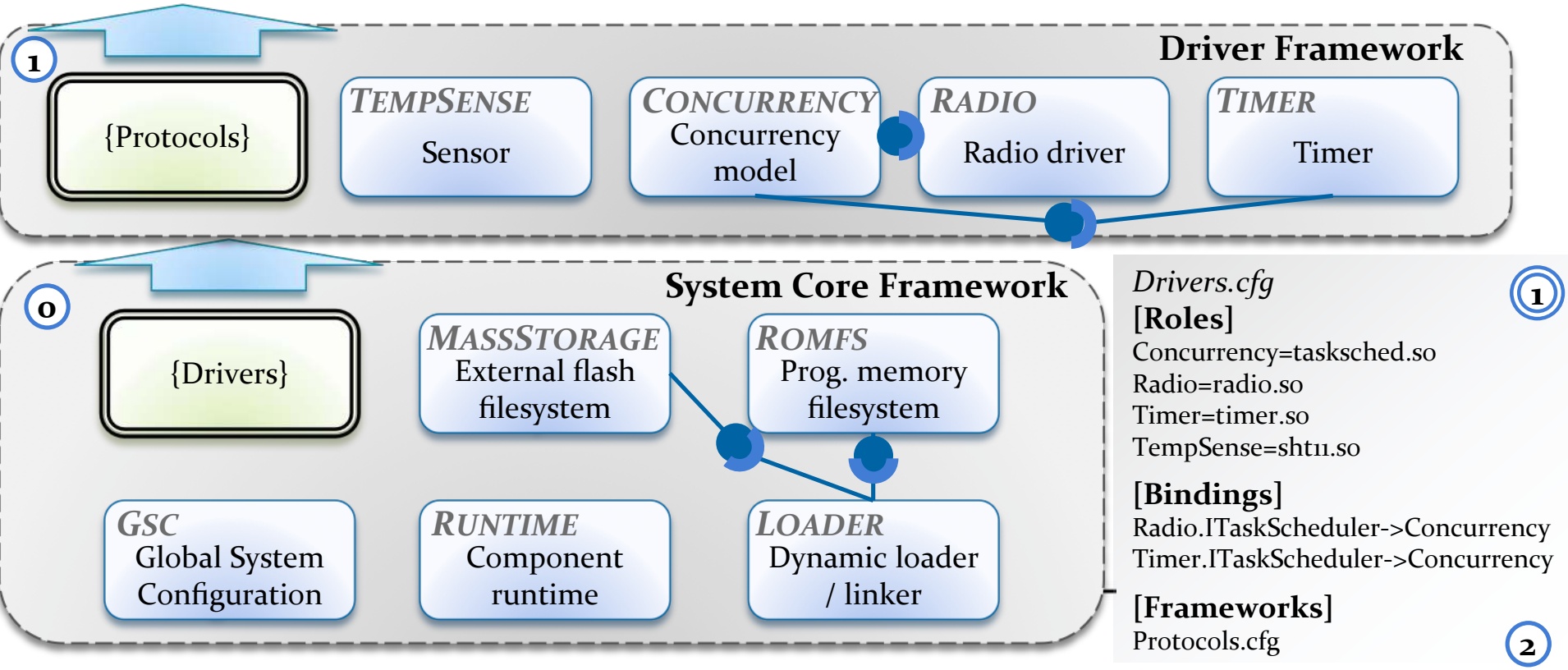
[Frameworks]

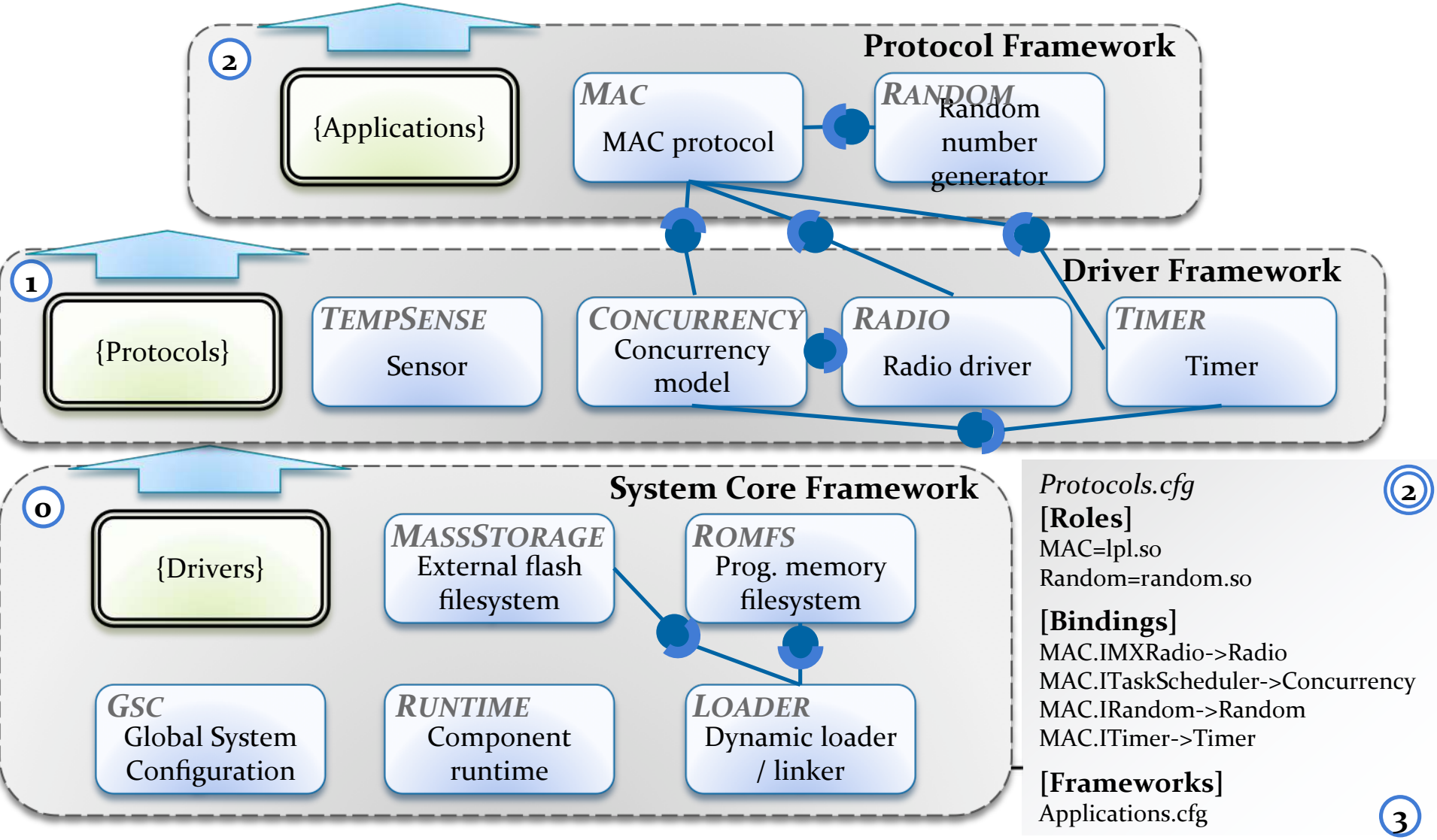
Drivers.cfg 1

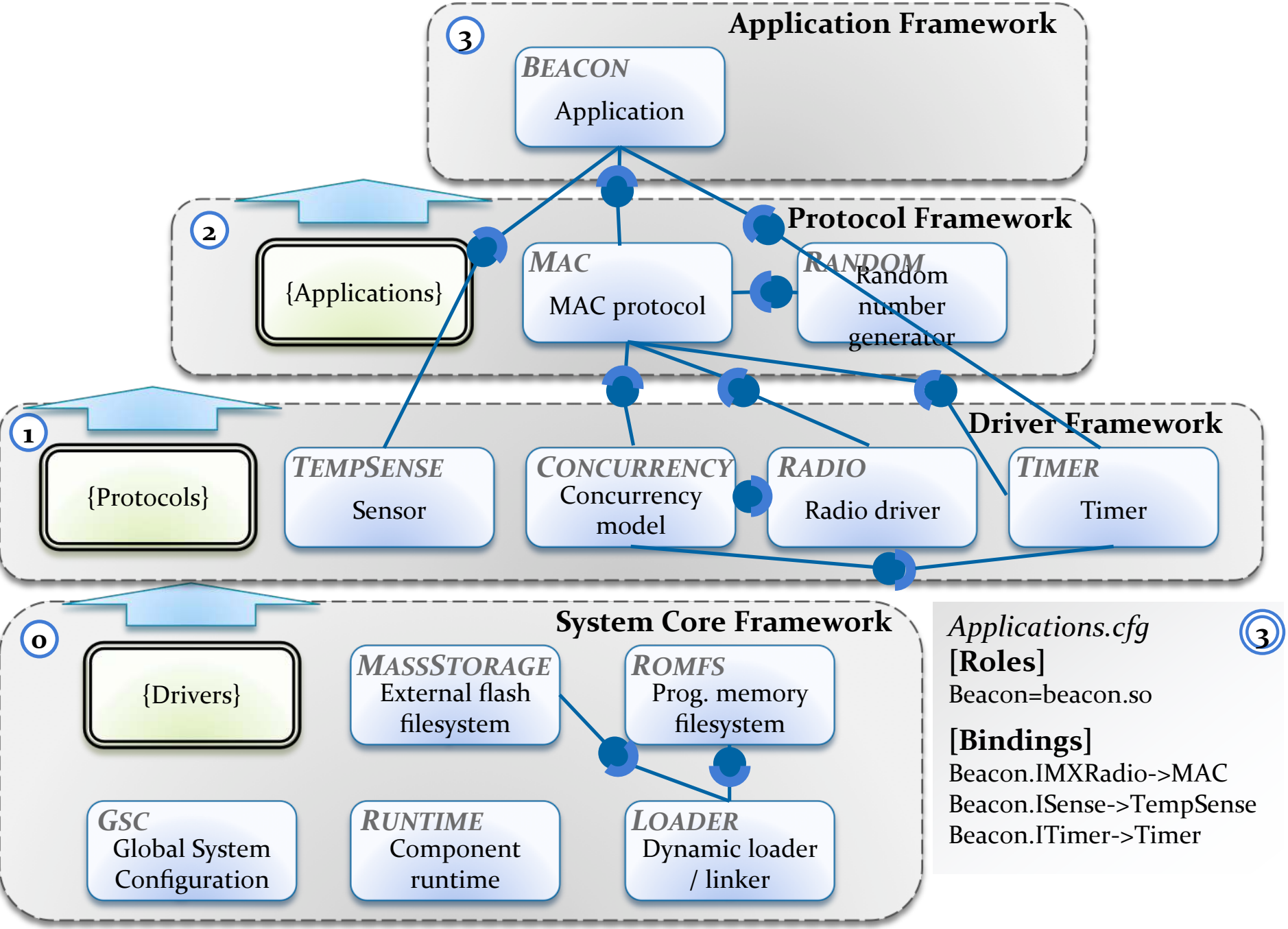
Concrete realisation on TelosB

- GSC, configuration files & roles

[48KB program memory, 10KB RAM]




2
Protocol Framework
{Applications}
MAC
MAC protocol
RANDOM
Random number generator
1
Driver Framework
{Protocols}
TEMPSENSE
Sensor
CONCURRENCY
Concurrency model
RADIO
Radio driver
TIMER
Timer
0
System Core Framework
{Drivers}
MASSSTORAGE
External flash filesystem
ROMFS
Prog. memory filesystem
GSC
Global System Configuration
RUNTIME
Component runtime
LOADER
Dynamic loader / linker
Protocols.cfg
2
[Roles]
MAC=lpl.so
Random=random.so
[Bindings]
MAC.IMXRadio->Radio
MAC.ITaskScheduler->Concurrency
MAC.IRandom->Random
MAC.ITimer->Timer
[Frameworks]
Applications.cfg
3


3
Application Framework
BEACON
Application
2
Protocol Framework
{Applications}
MAC
MAC protocol
RANDOM
Random number generator
1
Driver Framework
{Protocols}
TEMPSENSE
Sensor
CONCURRENCY
Concurrency model
RADIO
Radio driver
TIMER
Timer
0
System Core Framework
{Drivers}
MASSSTORAGE
External flash filesystem
ROMFS
Prog. memory filesystem
GSC
Global System Configuration
RUNTIME
Component runtime
LOADER
Dynamic loader / linker
Applications.cfg
3
[Roles]
Beacon=beacon.so
[Bindings]
Beacon.IMXRadio->MAC
Beacon.ISense->TempSense
Beacon.ITimer->Timer

Dynamic Reconfiguration

- Set of **architectural conventions**
  - → in particular layered acyclic architecture
  - → dedicated interfaces to trigger quiescent state
  - → state transfer interface (provided by implementer)
- **Primitives** MERGE, DETACH, EXTRACT and INJECT
  - → complete system reconfiguration capabilities

Applications.cfg
[Roles]
Beacon=beacon.so

[Bindings]
Beacon.IMXRadio->MAC
Beacon.ISense->TempSense
Beacon.ITimer->Timer

MERGE ←

MyNewApp.cfg
[Roles]
TimeSynch=timesynch.so

[Bindings]
TimeSynch.IMXRadio->MAC
TimeSynch.ITimer->Timer

=

Applications.cfg
[Roles]
Beacon=beacon.so
TimeSynch=timesynch.so

[Bindings]
Beacon.IMXRadio->MAC
Beacon.ISense->TempSense
Beacon.ITimer->Timer
TimeSynch.IMXRadio->MAC
TimeSynch.ITimer->Timer

Dynamic reconfiguration

- **Baseline**: TinyOS image-based update
  - ➔ both use Deluge (WSN reprogramming protocol)

Update description	**Data size (TinyOS)** [bytes]	**Data size (Lorien)** [bytes]	**Update time (TinyOS)** [ms]	**Update time (Lorien)** [ms]
Addition of application B	37606	1964	5816	992
Addition of applications B and C	37858	3327	5989	1504
Removal of application B *(post B/C addition)*	37606	165	5816	592
New version of random no. generator	37274	1122	5653	1584
New version of temperature sensor	37274	1443	5653	1565
New version of Deluge	37274	4387	5653	1968

Outline

Helping to realise distributed systems in:

- a **wide** world
- a **deep** world
- built (for and) **by humans**

Every hammer has two ends

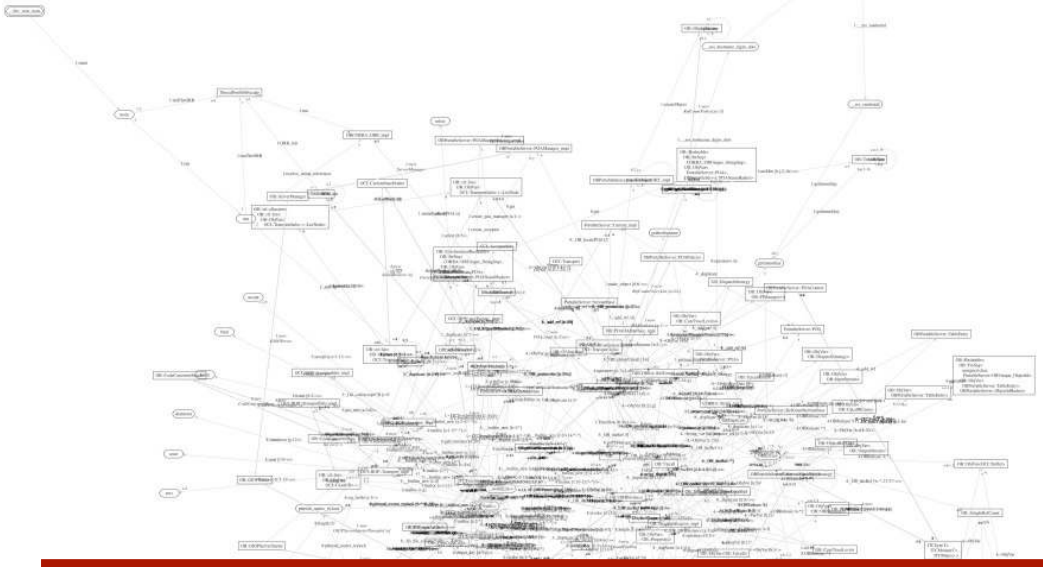
programmer (human)

abstractions (tool)

software (matter)

InfoLab21
Computing Department

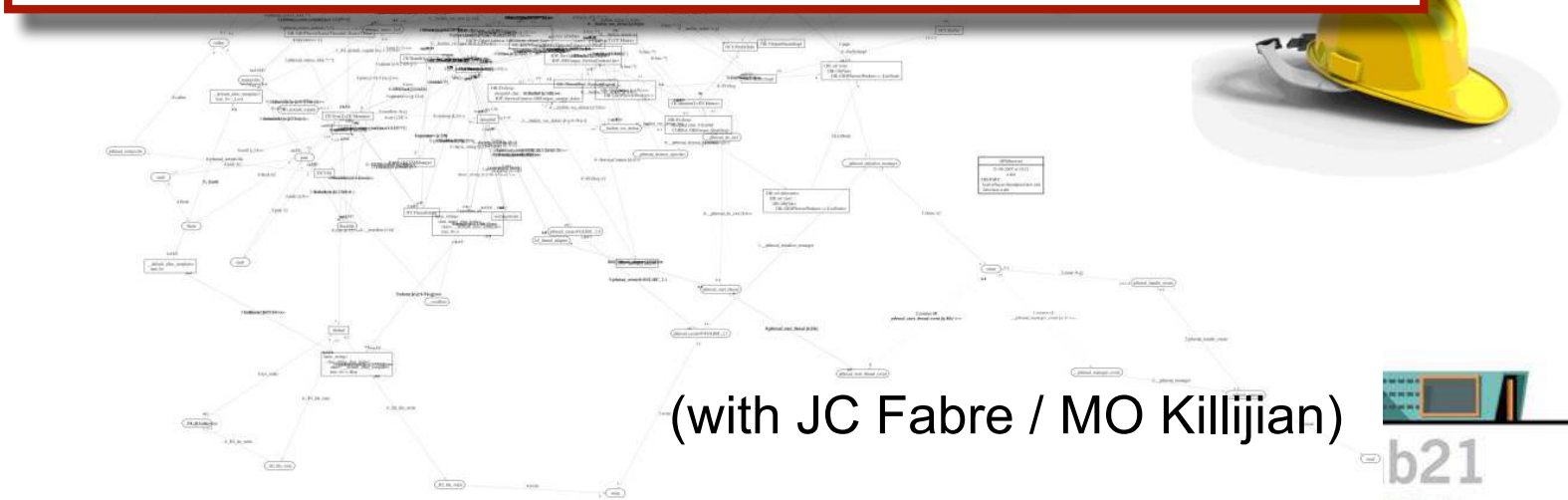
Example: recovering models



ORBacus Request Processing

one request,
2065 individual invocations,
over 50 C-functions and 140 C++ classes.

How to observe and analyse this?



(with JC Fabre / MO Killijian)

Approach: “interferometry”

- **Limit** observation costs: **more for less**
  - ➔ capture stack traces (more)
  - ➔ limit observation points to layer boundaries (less)
- **Filter** out unneeded data: “interferometry”
  - ➔ graph manipulation script language
  - ➔ reusable filters

vs.

Graph manipulation filters

Provides **variables** to store intermediary results

A number of graph **operators**

- **selection**

  `put ::pthread_create* G CREATE`

- **recursive extension**

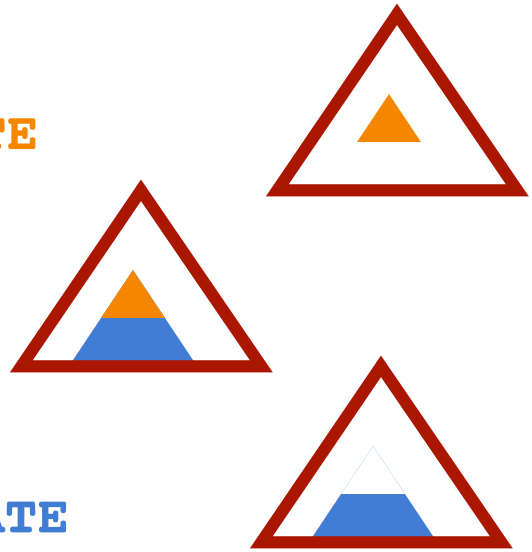
  `forward CREATE G`

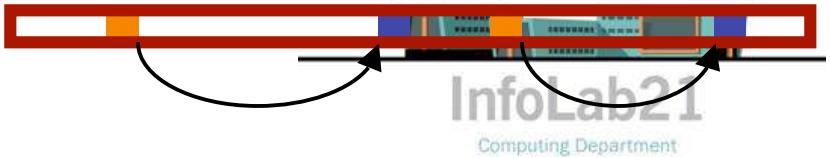
- **boolean algebra**

  `remove ::pthread_create* CREATE`

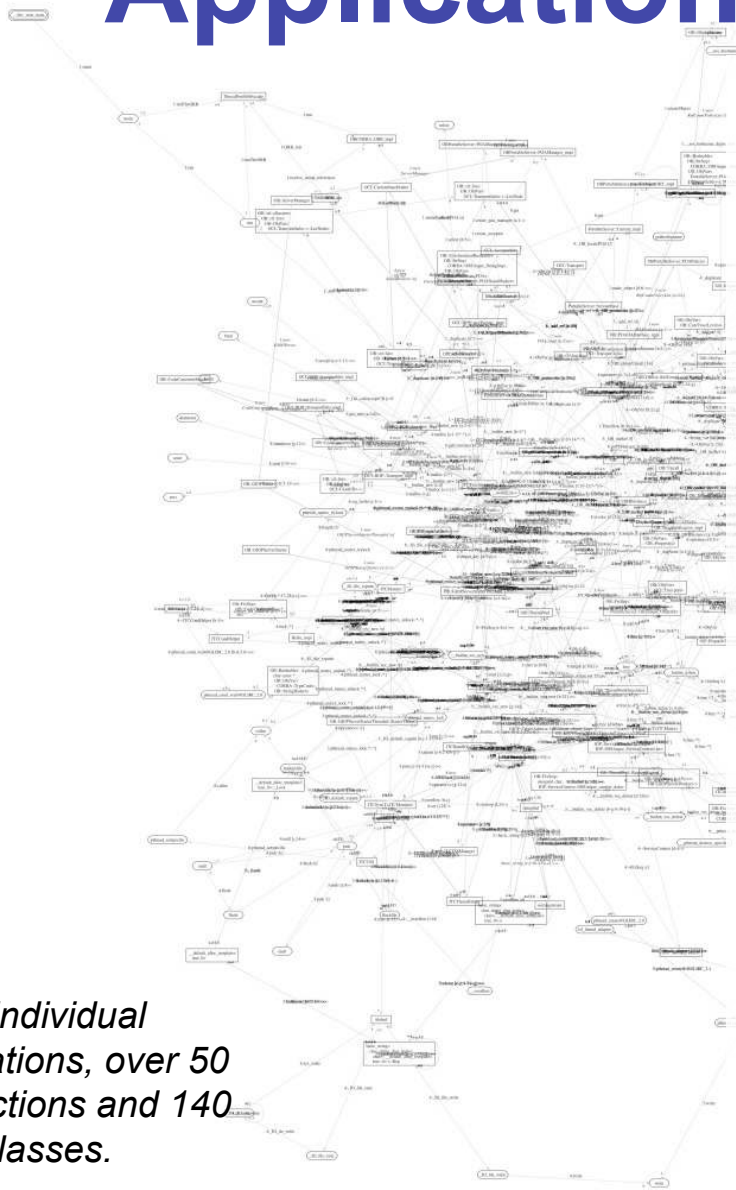
- **temporal operations**

  `fuse ::pthread_create* ::pthread_start_thread* G`





Application to Orbacus

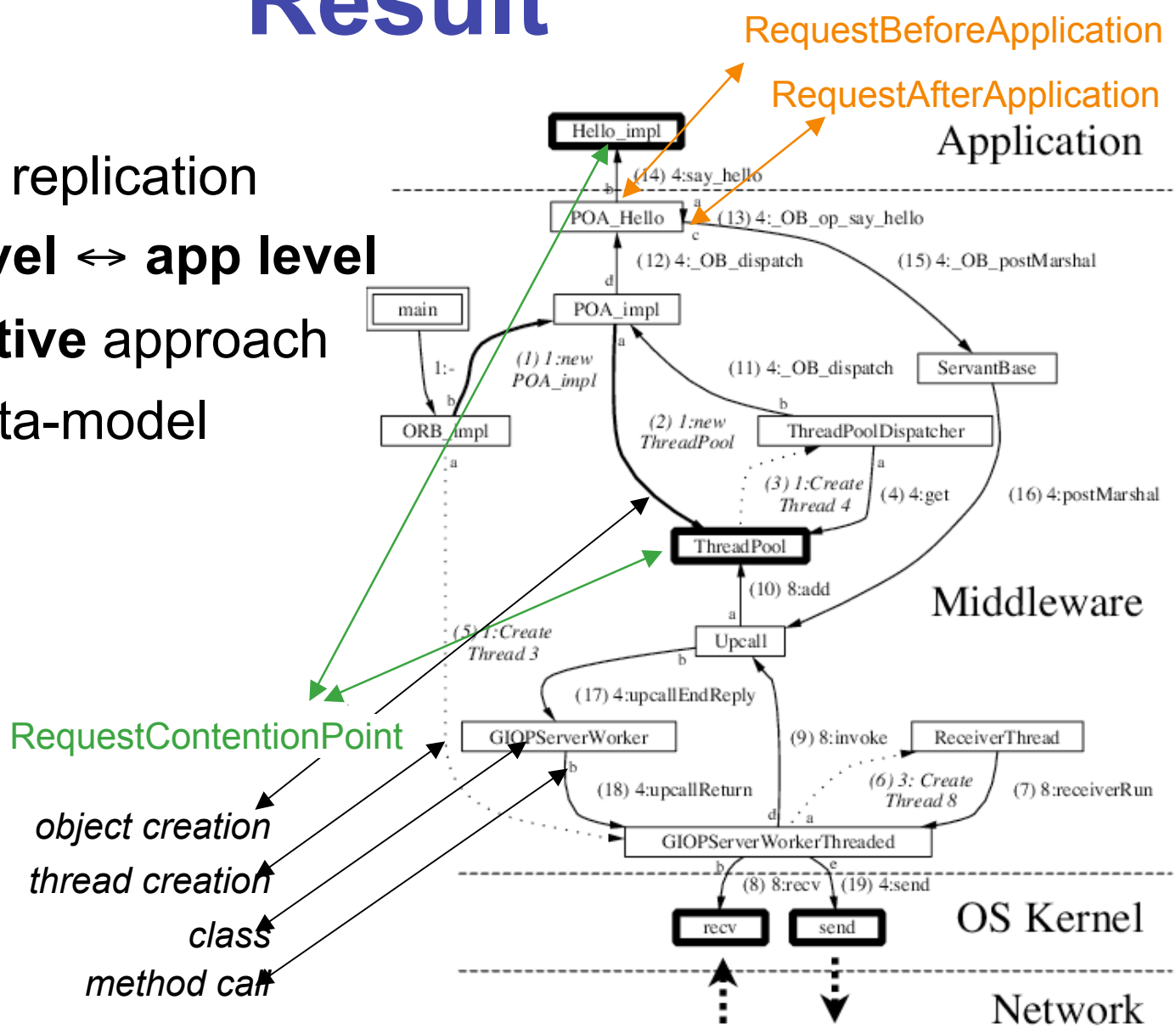


```
// [remove pthread]
put ::recv'* GlobalGraph R
put ::send'* GlobalGraph R
put Hello_impl::say_hello'* GlobalGraph R
put ::accept'* GlobalGraph R
backward R GlobalGraph
put ::lsf_thread_adapter'* ABS
put JTC* ABS
put OCI::* ABS
put ::__libc_start_main't1-0 ABS
put ::run't1-9 ABS
put OB::ORBControl::initializeRootPOA't1-11 ABS
put OBPortableServer::POAManagerFactory_impl::create_poa_manager't1-12 POA
forwN 3 POA R
add POA ABS
put OBPortableServer::POAPolicies::POAPolicies't1-42 ABS
put OB::DispatchStrategyFactory_impl::* ABS
abstract ABS R
abstract ABS GlobalGraph
remove ::accept't3-882 R
absPatern ::recv't8-828 R
absPatern ::recv't8-1309 R
absPatern ::recv't8-1908 R
put OB::Upcall::Upcall't8-1083 U
backward U GlobalGraph
add U R
put OB::ThreadPool::add't8-1224 TP
put OB::ThreadPool::get't4-1265 TP
backward TP GlobalGraph
add TP R
put OB::GIOPServerStarterThreaded::StarterThread::* ABS2
put OB::DispatchThreadPool_impl::* ABS2
put OB::DispatchRequest_impl::* ABS2
put OB::POAOAInterface_impl::* ABS2
abstract ABS2 R
absPatern OB::GIOPServerWorker::executeRequest't8-* R
```

2065 individual invocations, over 50 C-functions and 140 C++ classes.

Result

- Deterministic replication
  - ➔link **OS level** ↔ **app level**
  - ➔use **reflective** approach
  - ➔realise meta-model



Outline

Helping to realise distributed systems in:

- a **wide** world
- a **deep** world
- built (for and) **by humans**

conclusion

What I've left out

A wide world: Systems of Systems

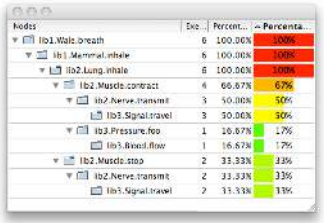
- Should evolve into another: **paradigm morphing**
  - ➔ combine **macroprogramming** & **components** (S. Lin)
- Supported by models: **models@runtime** (N. Bencomo)
  - ➔ model-driven engineering meets reflection

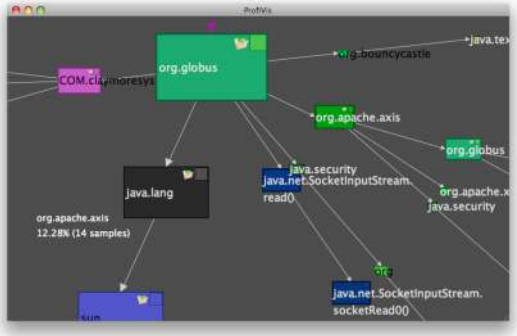
A deep world

- Flood monitoring using reconfigurable overlay framework

For humans

- anomaly comprehension [SoftViz 2010]

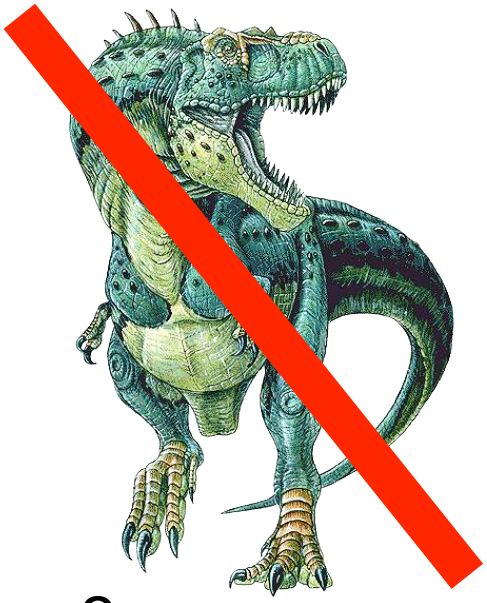




InfoLab21
Computing Department

Conclusion

- Many (most) of the works presented are **on-going**
- Trend: **The end of distributed programming** (*) ?
  - ➔ **declarative** (no behaviour)
  - ➔ **compositional** (no programming)
  - ➔ **macro-programming** (no distribution)
- Let's not forget developers are **humans**
  - ➔ every tool has two ends
  - ➔ and so do programming technologies
  - ➔ How do we analyse the fit of a technology?



*(as we know it?)

(Some) References

- **Interoperability**
- Valérie Issarny, Bernhard Steffen, Bengt Jonsson, Gordon Blair, Paul Grace, Marta Kwiatkowska , Radu Calinescu, Paola Inverardi, Massimo Tivoli, Antonia Bertolino, Antonino Sabetta, "CONNECT Challenges: Towards Emergent Connectors for Eternal Networked Systems", In Proceedings of the 14th IEEE International Conference on Engineering of Complex Computer Systems, Pottsdam, Germany, June 2009
- Carlos A. Flores-Cortés, Gordon S. Blair, Paul Grace, "An Adaptive Middleware to Overcome Service Discovery Heterogeneity in Mobile Ad Hoc Environments," IEEE Dist. Sys. Online, vol. 8, no. 7, pp. 1, 2007
- **Synergies**
- Exploiting Synergies Between Coexisting Overlays, Shen Lin, François Taïani, Gordon S. Blair, Proc. of the 9th IFIP Int. Conf. on Distr. App. and Interoperable Sys. (DAIS-2009), pp. 1-15 (14p)

(Some) References

- **WSN-software and embedded component-based proto-OS**
- Porter, B. & Coulson, G. Lorien: a pure dynamic component-based operating system for wireless sensor networks. MidSens '09: Proc. of the 4th Int. Workshop on Middleware Tools, Services and Run-Time Support for Sensor Networks, ACM, 2009, p. 7-12 (6p.)
- A generic component model for building systems software, Geoff Coulson, Gordon Blair, Paul Grace, François Taïani, Ackbar Joolia, Kevin Lee, Jo Ueyama, Thirunavukkarasu Sivaharan, ACM Transactions on Computer Systems (TOCS) 26(1): (2008), p.1-42 (42p)
- Experiences with Open Overlays: A Middleware Approach to Network Heterogeneity, Paul Grace, Danny Hughes, Barry Porter, Gordon Blair, Geoff Coulson, François Taïani, Proc. of the 3rd ACM SIGOPS/EuroSys European Conf. on Com. Syst. 2008 (Eurosys 2008)

(Some) References

- **Human comprehension**
- Towards anomaly comprehension: Using structural compression to navigate profiling call-trees, Shen Lin, Francois Taiani, Thomas C. Ormerod, Linden J. Ball, to be presented at the ACM Symp. on Soft. Visualization (SoftViz), Oct. 25-26, 2010 (10p.)
- CosmOpen: Dynamic reverse-engineering on a budget (journal version), François Taïani, Marc-Olivier Killijian, Jean-Charles Fabre, Software: Practice and Experience, John Wiley & Sons, Ltd. 39(18): (Dec. 2009) pp. 1467-1514 (48p.)

Thank you

Questions?