NETKIT : a "globally-applied" component-based approach to programmable networking

Geoff Coulson

Distributed Multimedia Research Group,
Department of Computing,
Lancaster University

geoff@comp.lancs.ac.uk

The NETKIT project

- started October 1st
- faculty
  - Geoff Coulson, Gordon Blair, David Hutchison
- PhD students/ researchers
  - Jo Ueyama, Kevin Lee, Ackbar Joolia, Irvin Ye
- building on
  - prior work on reflective, component-based, middleware (OpenCOM, OpenORB)
  - masters projects (Kevin Lee, Bholanath Surajbali)

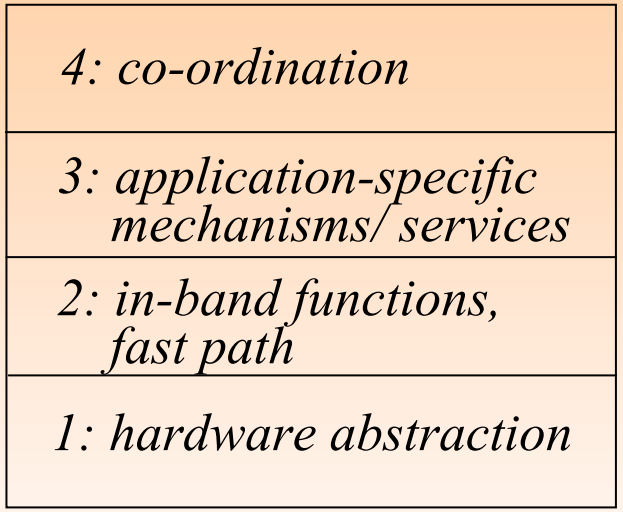
Acknowledgements: UK EPSRC; Intel Corp.

Motivation and context

- increasingly achnowledged demand for openness and programmability in networks
- e.g., need to dynamically introduce...
  - *mechanisms* like media-stream filters, security firewalls, QoS elements (e.g. intserv/ diffserv/ MPLS/ RSVP/ RED/ ECN)
  - *services* like dynamic private virtual networks, e-Science GRIDs, ubiquitous computing, ...
- must be able to *manage* mechanisms and services
  - *configure* (instantiate on routers; initialise and connect)
  - *reconfigure* (adapt; extend; evolve; remove)

Our perspective on the big picture

- stratum 4:
  - out-of-band signaling protocols, (e.g., RSVP; or dPVN co-ordination protocols)
- stratum 3:
  - 'programs' in the AN sense; act on pre-selected flows; not so performance critical
- stratum 2:
  - low-level, in-band, fine-grained, packet forwarding; highly performance critical
- stratum 1:
  - minimal OS-like support for higher-level programmability



Current paradigms in the light of this big picture

4: co-ordination **(open signaling; ALAN)**
3: application services **(AN; ALAN)**
2: in-band functions **(open signaling)**
1: hardware abstraction **(OSKIT? Scout? THINK?)**

- lack of integration?
- lack of generic support for *management* (configuration and reconfiguration)

Proposal: a globally-applied component-based approach

- two key aspects
  1. a *language*-, *platform*- and *paradigm*-independent component-based programming model
     - *uniformly applicable* in all four strata without unacceptable compromise (e.g. in terms of performance)
  2. built-in generic and flexible support for management
     - both configuration and reconfiguration of mechanisms and services in all strata

Three-pronged approach

component model

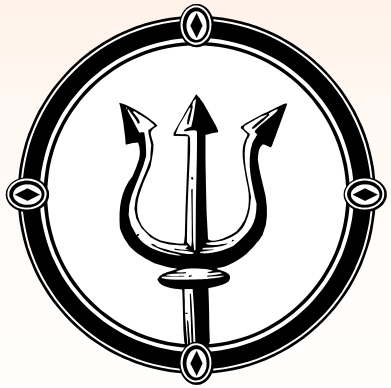
components everywhere

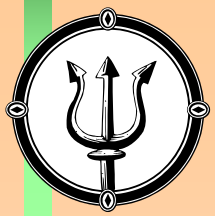
reflective meta-models

flexibility, openness

component frameworks

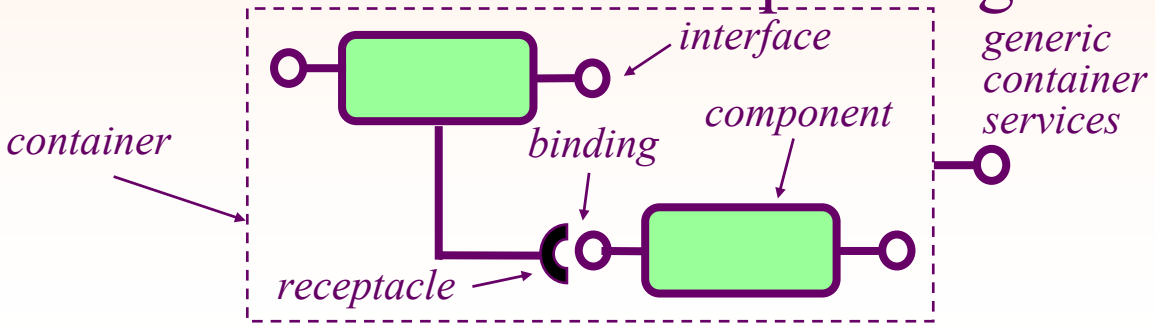
structure, constraint

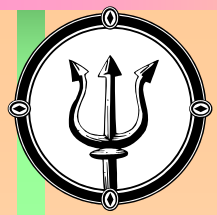




Prong 1: Components

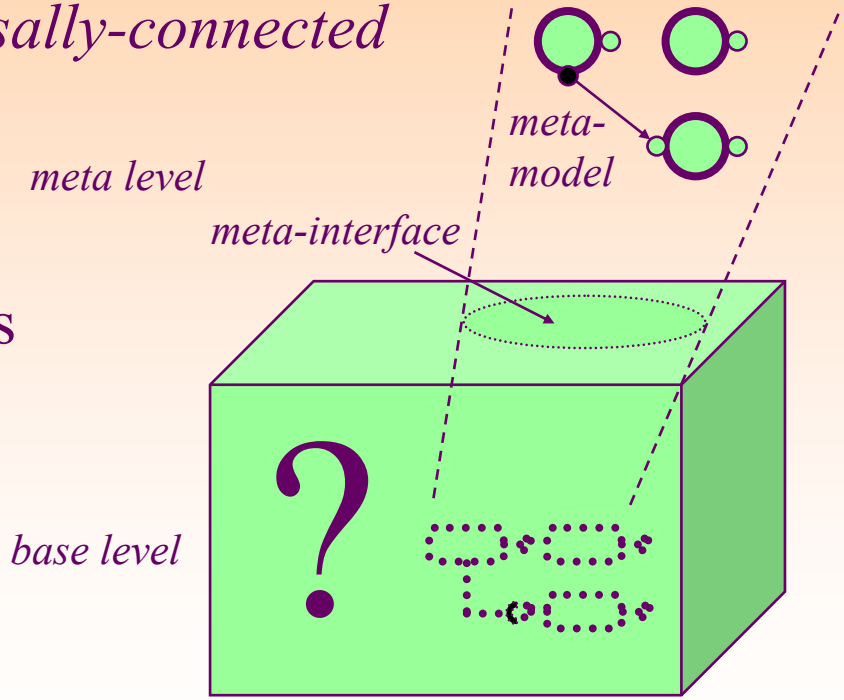
- lightweight, language-independent, *intra-container (i.e. address space)* model
  - 'binding' between 'receptacles' and 'interfaces' (receptacles render dependencies explicit)
    - build *inter*-container communication *on top*, as component-based middleware
- apply ubiquitously
- built-in reflective meta-models to help manage (next...)

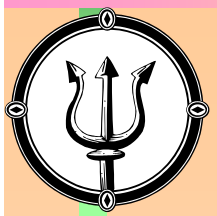




Prong 2: Reflective meta-models

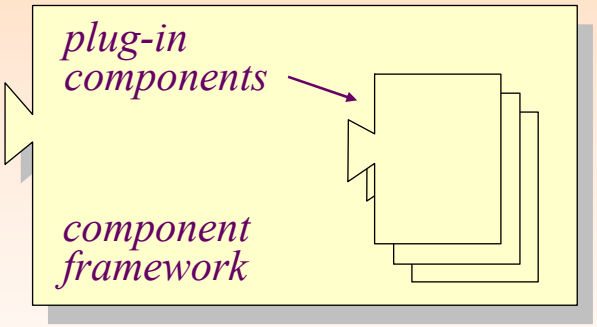
- reflection: look inside a “black box”—and see one or more self-representations (“meta-models”)
- meta-models are *causally-connected*
  - inspect
  - adapt
  - extend
- example meta-models
  - architecture
  - interface
  - interception
  - resource





Prong 3: Component frameworks

- *"CFs are collections of rules and interfaces that govern the interaction of components 'plugged into' them"*—Szyperski
- domain-specific 'life-support environments' for plug-in components
- examples
  - stratum 2: 'frameworks' à la Click, LARA++
  - stratum 3: CF-based execution environments
  - stratum 4: CF-based middleware-mediated signaling (OpenORB)



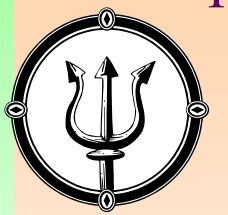
Implementation to date

- OpenCOM: a lightweight component model
  - based on superset of minimal subset of MS COM
  - features built-in reflective meta-models
- OpenORB: a CF-based reflective middleware platform (for stratum 4)
  - highly configurable and reconfigurable
  - extends OpenCOM's reflective meta-models to the distributed case
- OpenCOM-based CFs for PC routers, media filters

Upcoming work

- OpenCOM enhancements
  - port to Linux/80*86, Linux/StrongARM, StrongARM, and IXP1200 micro-engines
  - performance (via ‘code rewriting’), co-existence of compiled code and bytecode, basic security mechanisms
- Case-study based on Columbia’s Genesis Kernel
  - => development of an extensible component library; CFs in other strata; incorporate external code; reconfiguration experiments
- dependability issues: e.g., integrity, safety and security

Conclusions

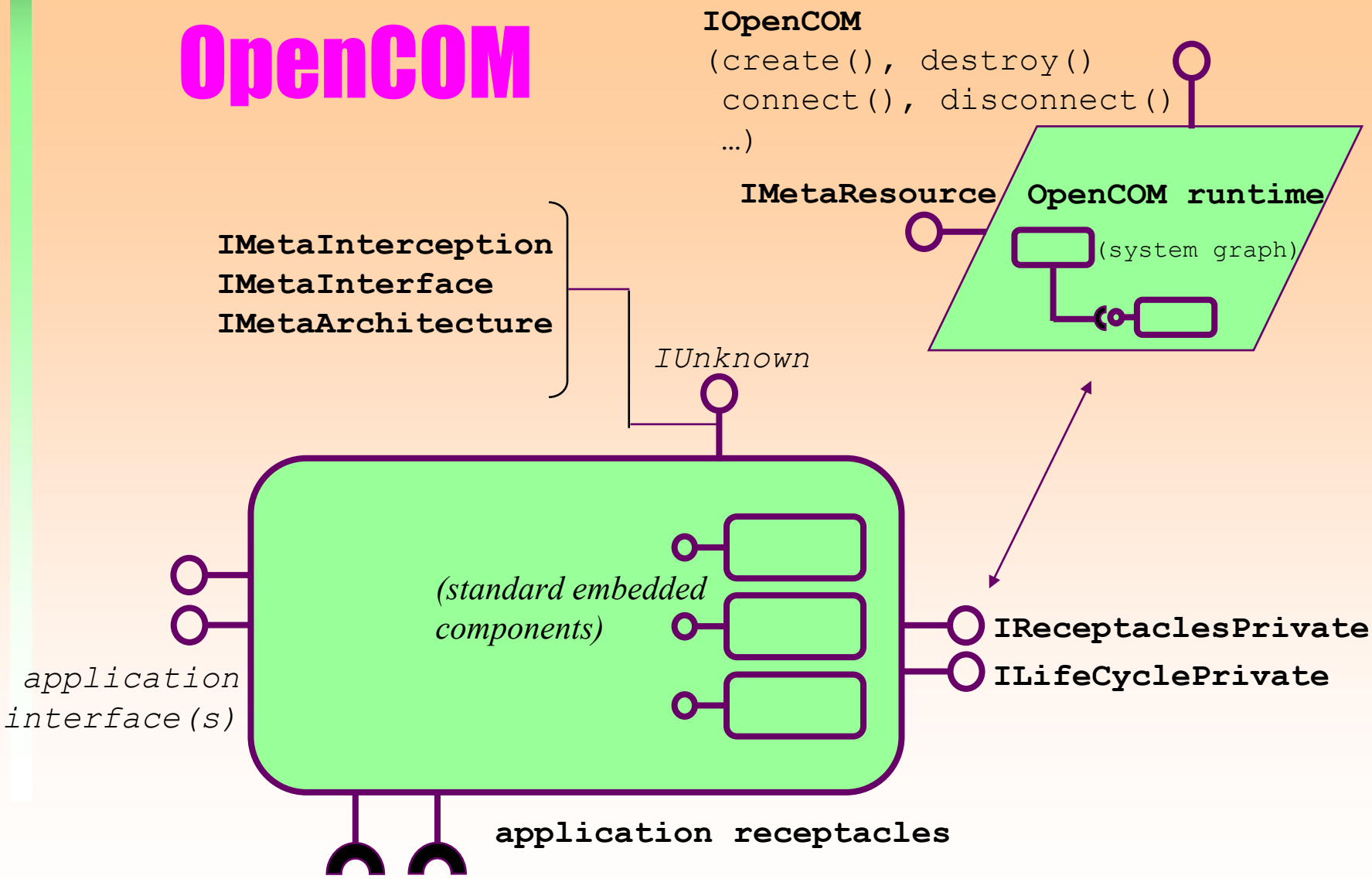


- we propose a globally component-based approach to programmable networking
  - uniform programming model (components everywhere)
  - reflective meta-models (flexibility, openness)
  - component frameworks (structure, constraint)
- approach already validated in reflective middleware environment
- potential benefits
  - vertical integration
  - language and system independence
  - fundamental support for the management of configuration and reconfiguration
  - support for multiple programmable networking paradigms, and framework for integration of existing techniques, software, ...

OpenCOM: a lightweight component model

- a superset of a subset of Microsoft's COM
- we take...
  - efficiency (*vtable* standard)
  - dynamic load/ unload
  - language-independence
  - interoperability with COM world
- we add...
  - dependency tracking through explicit receptacles
  - the four reflective meta-models

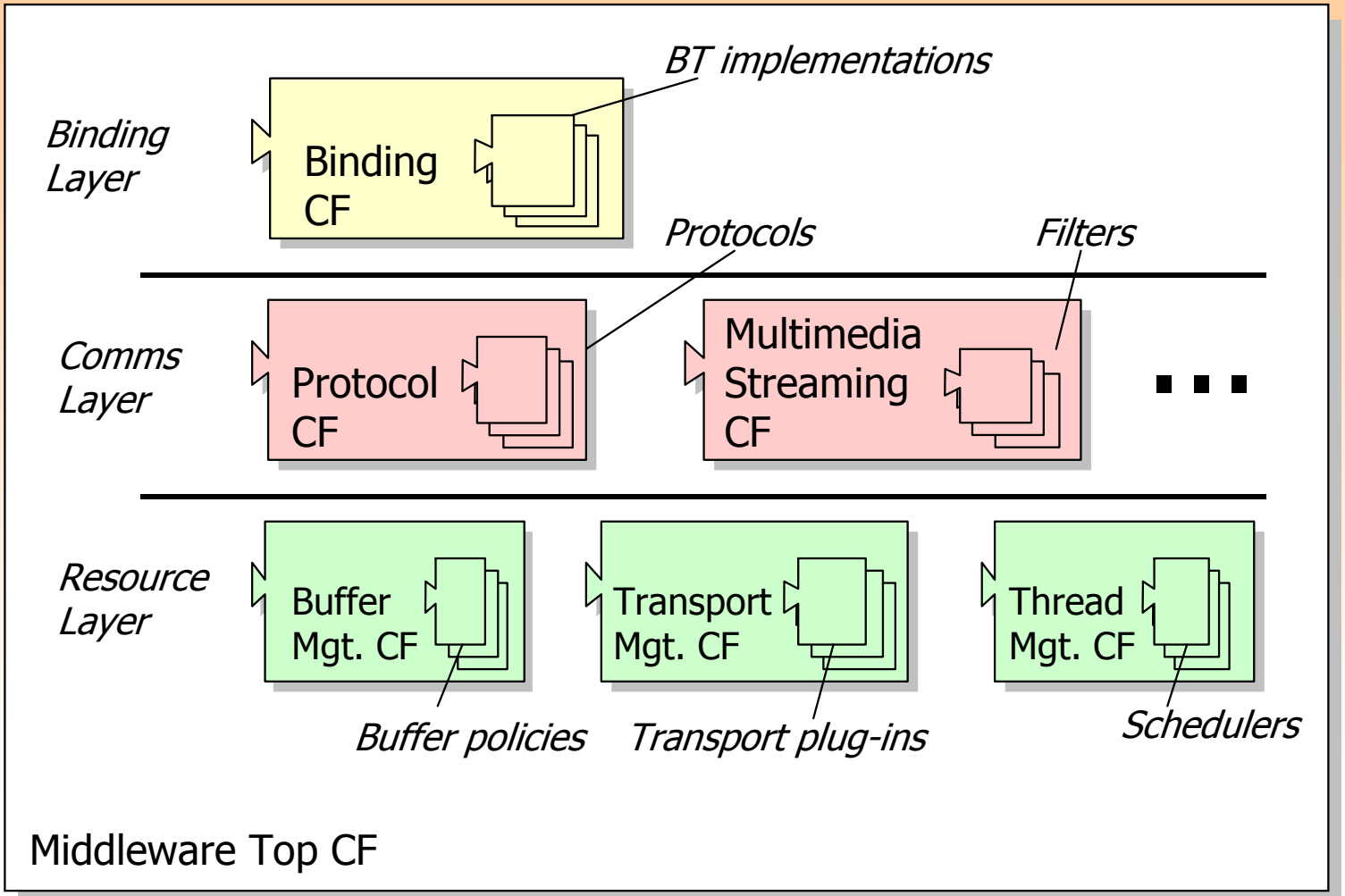
OpenCOM


IOpenCOM
(create(), destroy()
connect(), disconnect()
…)
IMetaResource
OpenCOM runtime
(system graph)
IMetaInterception
IMetaInterface
IMetaArchitecture
IUnknown
(standard embedded components)
IReceptaclesPrivate
ILifeCyclePrivate
application interface(s)
application receptacles

OpenORB v2: a CF-based reflective middleware platform

- built using OpenCOM in C++
- structured using component frameworks
- can configure and reconfigure a range of API 'personalities'
  - CORBA
  - RM-ODP based media-streaming
  - SOAP-based Web Services
  - etc.

The structure of OpenORB v2



The meta-models...