

# Advanced Topics in Distributed Systems

INF5090

## Autonomic Networking

Computing Department  
 University of Oslo

25<sup>th</sup> March 2007

Dr. Andreas Mauthe, andreas@comp.lancs.ac.uk

# Contents

- **Outline**
  - Motivation
    - Why do we need a new networking paradigm
    - The drawbacks of the IP hourglass model
    - The self-x attributes
  - Self-organisation
    - Example of network view
  - Autonomic Networking Architecture
    - ANA node architecture
    - ANA network architecture
  - Autonomic Networking Concepts
    - Compartments
    - Naming and addressing
  - Functional Composition
    - Functional model
    - Outline
    - Evaluation
  - Monitoring
    - Monitoring concepts
    - Basic architecture
  - ANA basic design concepts
    - MINMEX
    - API
    - Platforms
- **Objectives**
  - You should become familiar with the ideas behind autonomic networking
  - You should know about the main aspects of the Autonomic Networking Architecture
    - Functional composition
    - Monitoring
  - You should have an overview of the basic design and implementation concepts

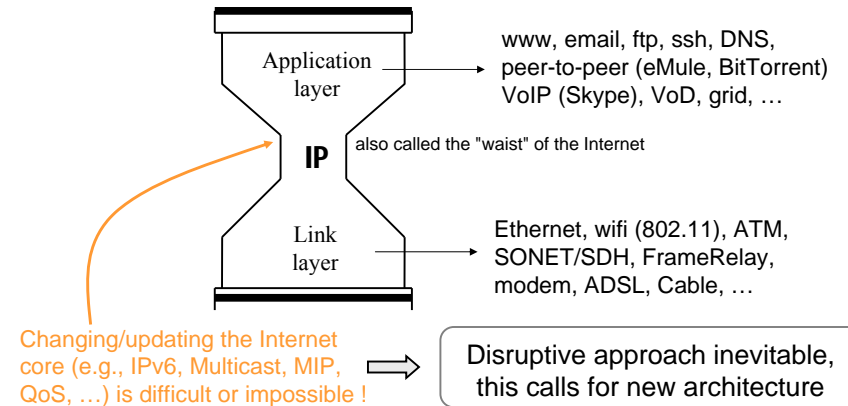
## Motivation (I)

Paradigm shift in Communication – is it required and if yes why?

- **The Internet suffers from architectural stress:**
  - Not ready to integrate and manage the envisaged huge numbers of dynamically attached devices (wireless revolution, mobility, personal area networks, etc.)
  - Does not optimally support different forms of communication
    - Only packets switched
    - No QoS, etc.
  - Lacks integrated monitoring and security mechanisms
- **Issues**
  - No functional scalability
  - Increasing costs of network management
  - New services required
  - Security threads
- **Goal**
  - Specifications, architectures and techniques that make networks more scalable, adaptive and autonomic

## Motivation (II)

- **Variability in the Internet is above and below IP:**  
 → the "hour-glass" model



## Challenges and Approach



- **Design and develop New Networking Principles**
- **From static networks to flexible network structures**
- **From hierarchical routing to routing between network compartments**
- **Allow for interconnecting heterogeneous networks**
- **Ease network management – add autonomic features**
- **Develop a new Network Node Design**
- **From static layers to flexible compartments**
- **Multiple node compartments run in parallel**
- **From static layers to functional composition**
  - Use functions when needed
- **Include node and network monitoring**



## Characteristics of Autonomic Networks



- **Autonomicity of network nodes**
    - Features: auto-configuring, operation independence, self-managing
    - Nodes are not managed by external entities
  - **Scalability**
    - Physical scalability
      - In terms of number of network nodes and communication entities
    - Functional scalability
      - Providing adequate functionality for different network types
        - e.g. small wireless ad-hoc networks to global high-speed networks
  - **Adaptability**
    - Network conditions
      - Changes in workload, resource availability, etc.
    - Exceptional circumstances
      - Failures, attacks, etc.
  - **Simplicity**
    - In development and deployment
- ➔ **Autonomous networks and their components should require little or no direct intervention during set-up and runtime but still provide a stable, reliable and secure communication infrastructure adapted to the environment they operate in and the requirements of the applications**
- Autonomic networks are autonomous networks with the ability to learn and adapt to changes in the environment



## The Self-x Attributes



- **Fundamental autonomic networking principles are expressed in various self-x attributes**
  - **Self-organising**
    - **Network nodes organise themselves to form a community**
      - Dynamic role assignment
      - Joint decision making
  - **Self-managing**
    - **Network nodes manage their behaviour according to context and rules**
    - **Self-configuring**
      - First step of self-management within an autonomous network
  - **Self-optimising**
    - **Network nodes**
      - Adaptation of node behaviour to regular network conditions
    - **Network**
      - Global optimisation through joint decision making
  - **Self-monitoring**
    - **Network nodes monitor their own state and the network state**
      - Autonomous information sensing and processing
      - Observation of neighbour behaviour
  - **Self-healing**
    - Networks can recover from node failures through re-organisation
    - Nodes can recover through re-configuration
  - **Self-protection**
    - Resilience against attacks and male-behaviour



## Distributed Decision Making

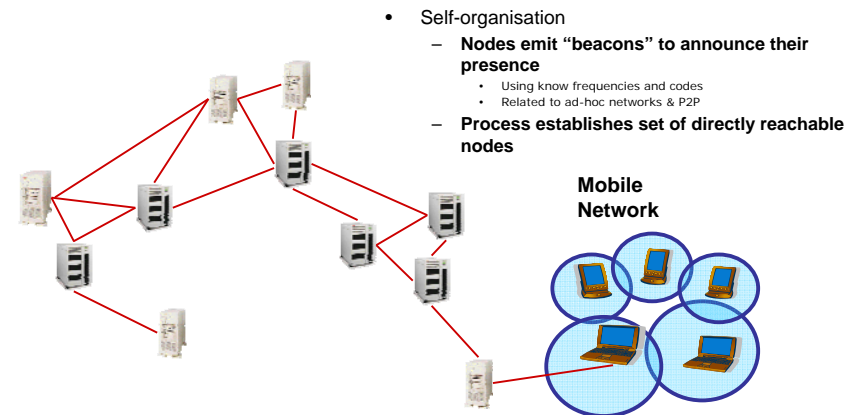
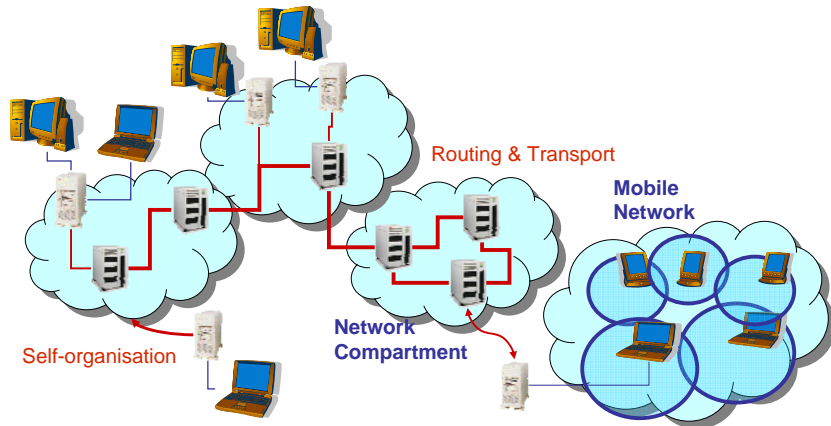


- **Decisions are taken in the network**
  - Forwarding, multicast, filtering, translating, etc.
  - What kind of decisions can be taken in the Network?
    - How about ...
      - Blocking packets, flow prioritisation, encryption, etc.
- **Who is taking decisions?**
  - Network nodes
    - Collectively or individual
  - Based on
    - **Situation/ context awareness**
      - (local) knowledge about the situation
      - Constantly evolving
      - Levels:
        - Perception: perceiving (critical) factors in the environment
        - Inference: understanding what those factors imply
        - Prediction: predicting system future state
    - **Information exchange with other nodes**
    - **Policies**
      - Locally executed
      - Must result in co-ordinated behaviour
- **Issues**
  - Discovering misbehaviour
  - Reacting to misbehaviour
  - Trust and collaboration
  - Policy representation and compliance



- **Distributed information gathering**
  - Integral part of network nodes
  - Selective perception
- **Data selection**
  - Locally at network nodes
    - Selection and aggregation as early as possible
    - Configurable according to network and communication type
    - Measurement methods:
      - Active vs. passive
      - Off-line vs. in-line [Pezaros et al 2004]
      - Using IETF measurement protocols!?!?
        - IP Measurement Protocol (IPMP), One-Way Active Measurement Protocol (OWAMP), etc.
  - Capturing relevant data
- **Information exchange**
  - Controlled by decision process
    - Dynamic perception rules
  - Using IETF exchange protocols?!?
    - IP Flow Information eXport, PSAMP, etc.

- **Network formation**
  - Establishment of network structure and connectivity
    - Negotiation & auto-configuration of (fault tolerant) protocol components, protocol functions & network services
    - Self-organisation into resilient survivable networks
      - ➔ Infrastructures, protocols and signalling must be:
        - Secure and attack resistant
        - Authenticated if necessary
        - Adaptive and suitable for the deployment environment
    - Existing infrastructure should be used but not relied upon
      - Name servers, PKI, etc.
- **Network maintenance**
  - Autonomic operations to maintain the network
    - Self-managed
    - Self-diagnosis, & repair
    - Continuous re-optimisation

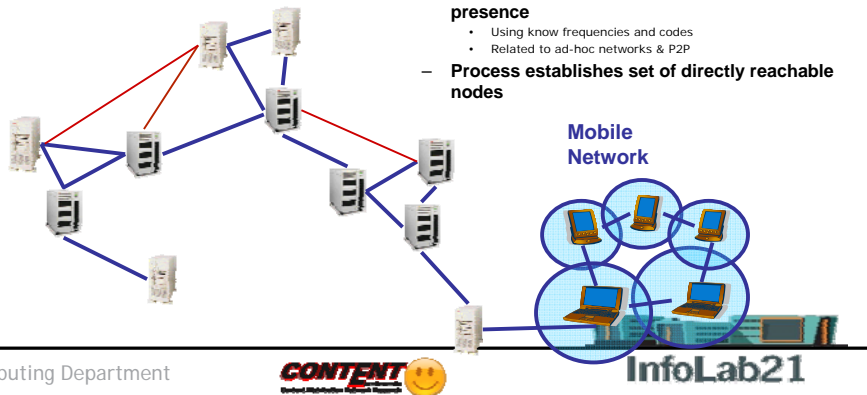


## Network Organisation: Self-Organisation (III)



- Link formation
  - **Pair-wise negotiation of link formation**
    - Interested nodes answer beacons
    - Exchange identification, node and link characteristics
      - Layer 2 connectivity structure
  - **Maintenance**
    - Keep alive messages

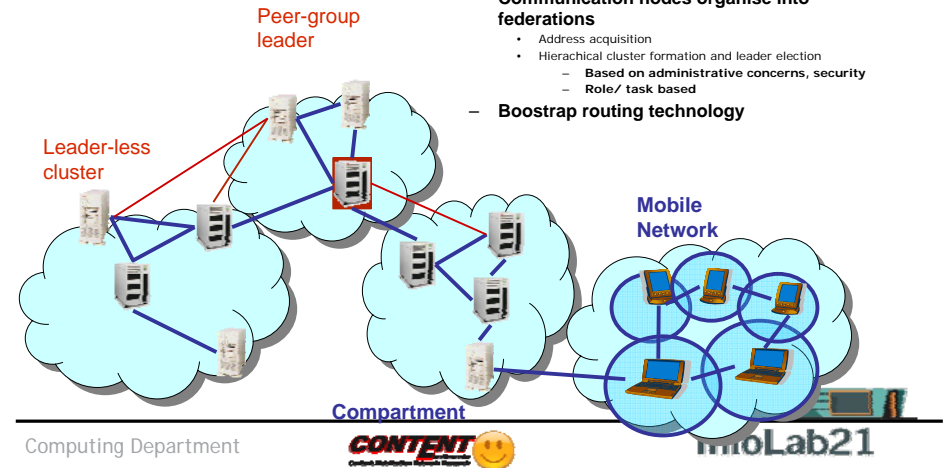
- Self-organisation
  - **Nodes emit "beacons" to announce their presence**
    - Using known frequencies and codes
    - Related to ad-hoc networks & P2P
  - **Process establishes set of directly reachable nodes**



## Network Organisation: Self-Organisation (IV)



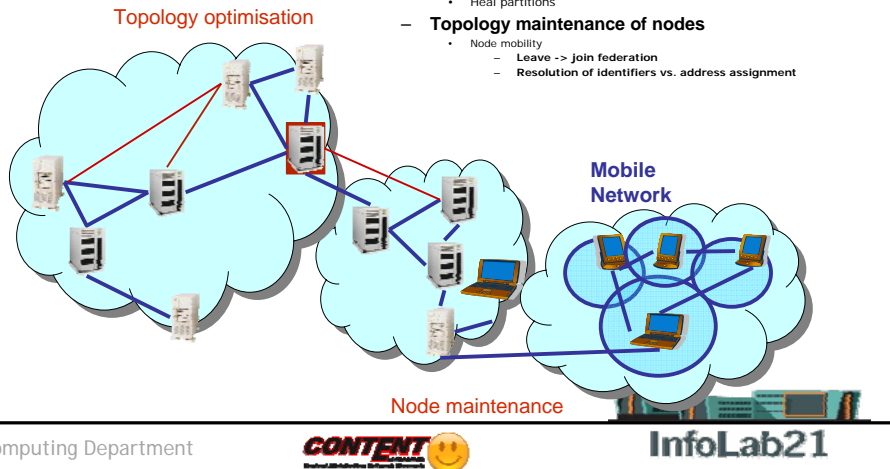
- Self-organisation & Federation
  - **Communication nodes organise into federations**
    - Address acquisition
    - Hierarchical cluster formation and leader election
      - Based on administrative concerns, security
      - Role/ task based
  - **Bootstrap routing technology**



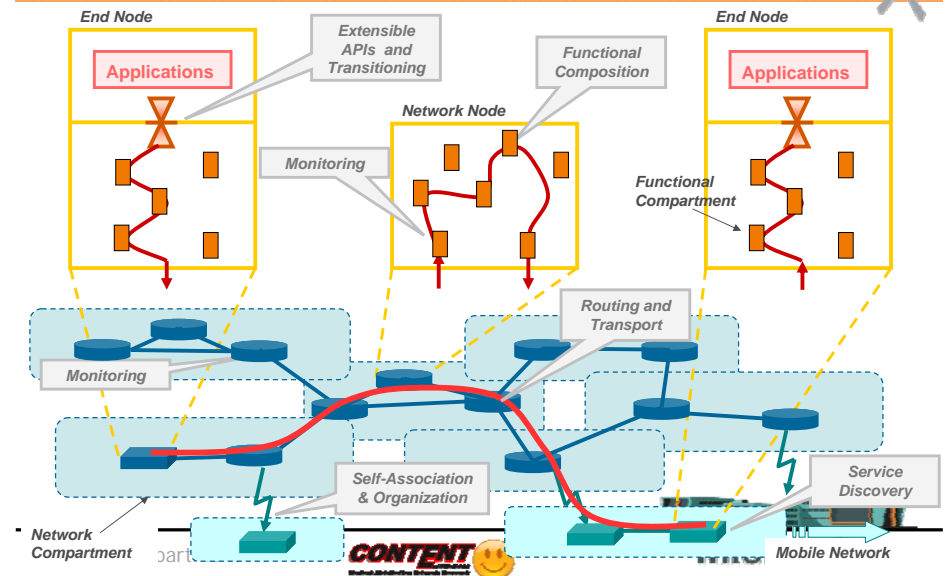
## Network Organisation: Self-Organisation (V)



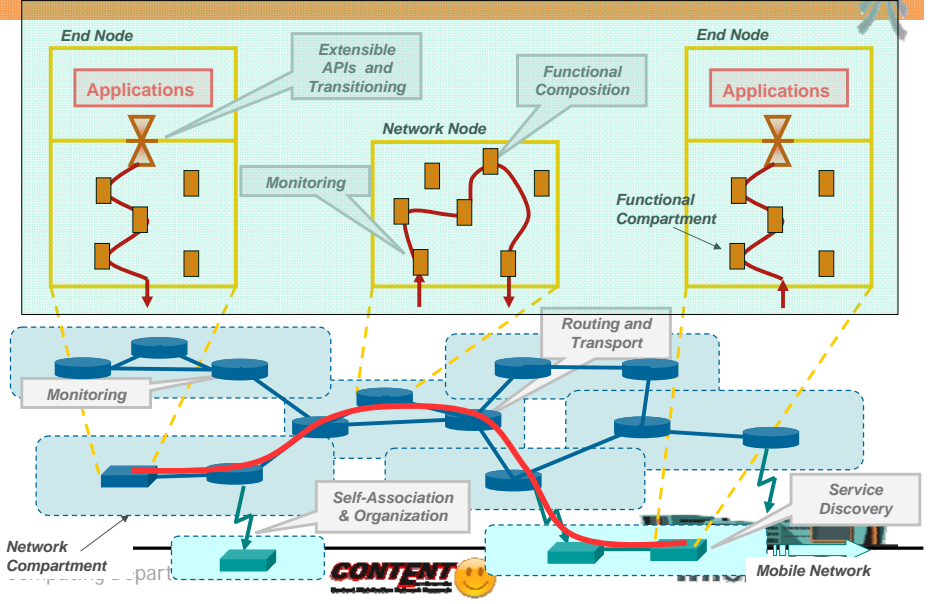
- Topology optimisation and maintenance
  - **Topology maintenance of federation**
    - Split/ merge
      - Due to group mobility and dynamic coalitions
    - Heal partitions
  - **Topology maintenance of nodes**
    - Node mobility
      - Leave -> Join federation
      - Resolution of identifiers vs. address assignment



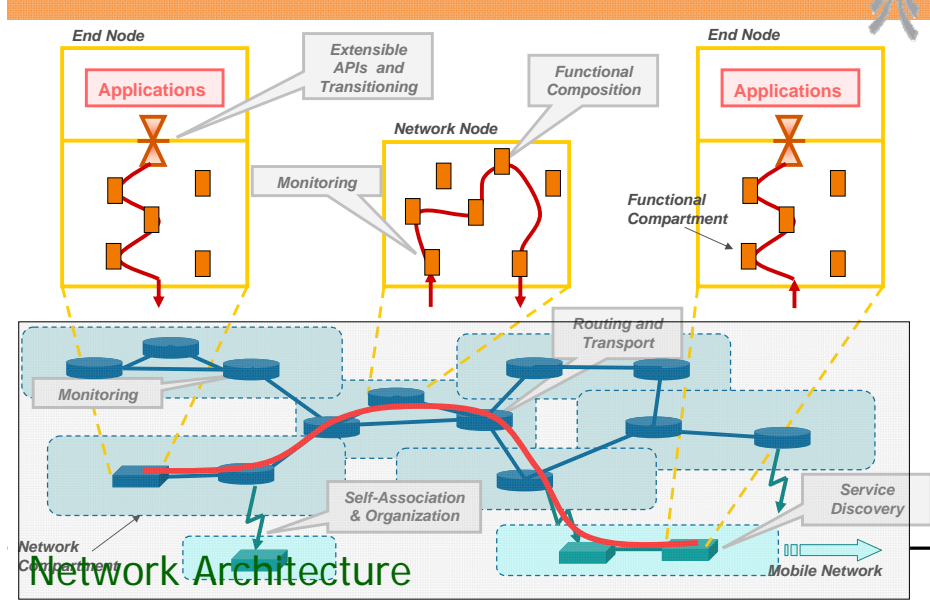
## The ANA Project View



## ANA Project View - Node Achitecture



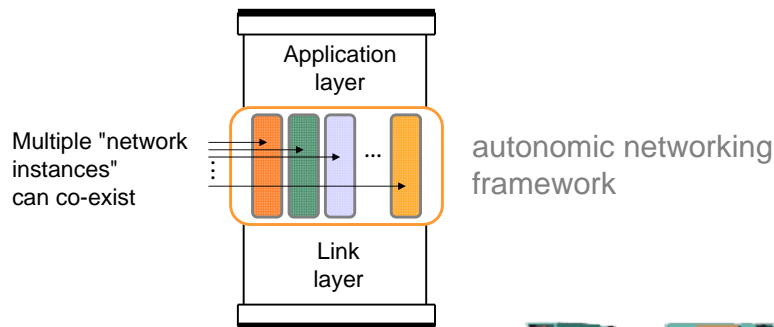
## ANA Project View - Network Architecture



## Autonomic Networking vs Internet



- "one-size-fits-all network waist" vs. adapted network instances.
  - ANA is a meta-architecture to host, interconnect, and federate multiple heterogeneous networks



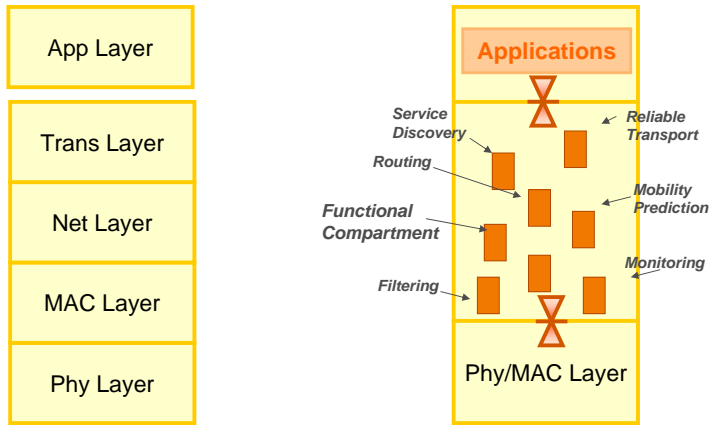
## Autonomic Networking Abstractions



- Abstractions (to be detailed in the following slides\*):
  - Compartment
  - Information Channel (IC)
  - Information Dispatch Point (IDP)
  - Functional Block (FB)

\* The slides are part of the ANA Blueprint



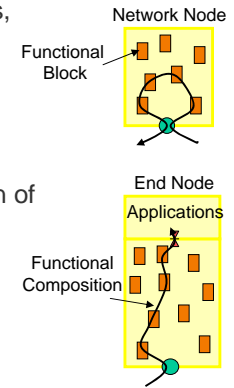


- **Organisation of network functions:**

- Composition of stack, interaction models, protocol layer fusion, interlayer control loops, cross layer optimisations
- Dynamic run-time deployment of network functions components (active networks)

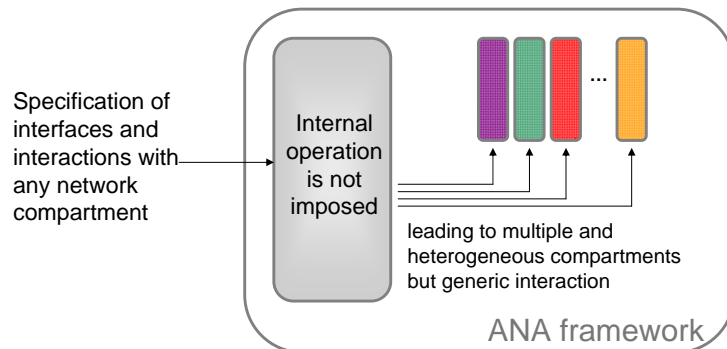
- **Self-Learning**

- No basic AI research in ANA, but application of machine learning to improve self-awareness of networks
- Applications in ANA: mining monitored data (anomaly detection), learning failures causes, detecting and discriminating traffic anomalies, predicting mobility



- **Compartment = wrapper for networks**

- Does not specify the internal structure but how compartments interact



- **A (network) compartment implements the operational rules and administrative policies for a given communication context. It defines:**

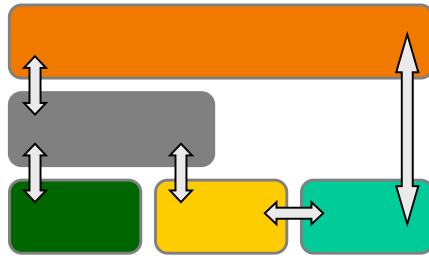
- How to join and leave a compartment: member registration, trust model, authentication, etc.
- How to reach (communicate with) another member: peer resolution, addressing, routing, etc.
- The compartment-wide policies: interaction rules with "external world", the compartment boundaries (administrative or technical), peerings with other compartments, etc.

➔ **Compartments decompose communication systems and networks into smaller and easier manageable units**

## Compartment Abstraction

- The compartment abstraction serves as the unit for the federation of networks into global-scale communication systems.

- Compartments can be overlaid, i.e. compartments can use the communication services of other compartments (and vice versa).



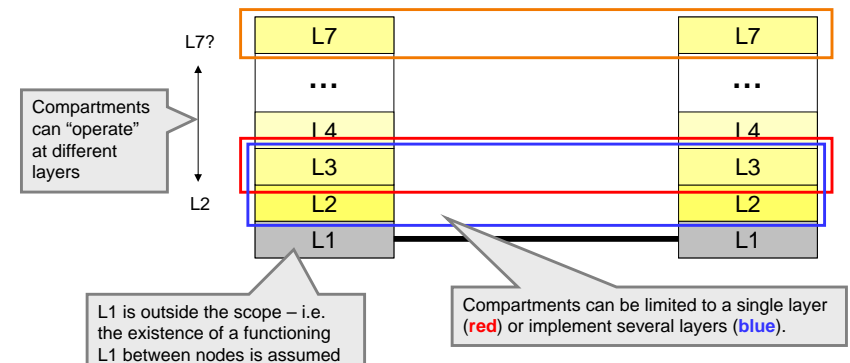
## Compartments Types

- **Network Compartment:**
  - Has membership predicate (i.e. members know whether they are part of a compartment or not)
    - Set of members (nodes)
  - Defines how intra-compartment naming/addressing, routing, forwarding, etc. works (depending on the compartment type, only some of these are required)
  - Defines a trust boundary
    - Imposes administrative policies
  - (Optional) Provides inter-compartment communication interface
- **Minimal ANA Compartment - as above, plus:**
  - Provides "MINMEX support"
    - Does this imply auto-configuration and self-organizing capabilities?
  - Provides signalling interface, e.g.:
    - query compartment information,
    - access control
    - join/leave procedure,
    - resolution process, ...
- **ANA Compartment - as above, plus:**
  - autonomic
    - self-\* properties
  - resilient & secure

## Example Compartments

- **Network Compartments**
  - L2 Network (e.g. Ethernet)
  - IPv4/IPv6 Internet, Autonomous Systems, ...
  - Corporate Network, Home Network, HotSpot ...
  - MobileIP, HIP, ...
  - DNS (this only provides a name space plus registration/resolution mechanism, i.e. is not really involved in data plane operation), ...
- **Minimal ANA Compartments**
  - Compartments that implementing MinMEX Framework and provide the default/minimum signalling interface
- **ANA Compartments**
  - Self-organizing/Self-optimizing Network (automatic, aware, adaptive)
  - ...

## Compartments vs. Layers



**Invariant:** Either all network entities along the e2e paths between the compartment members have to be part of the same compartment (and the compartment implements all required layers) or the compartment relies on an underlying communication layer or "underlay" compartment(s).

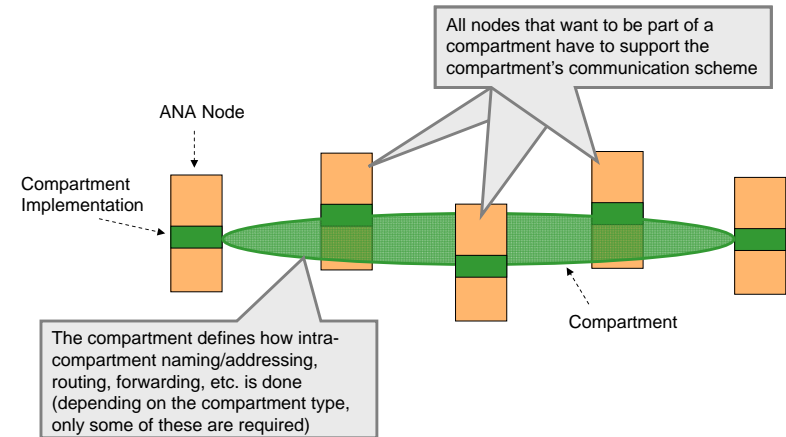
# Compartments vs. Layers



- **Compartments can be combined in more flexible ways than OSI-like layers (due to relaxation of the strict layer concept); e.g.:**
  - layers can be missed out ("L4"-Compartment over "L2"-Compartment)
  - layers can be repeated ("L3"-Compartment-over-"L3"-Compartment)
  - new "layers" (i.e. compartments) can be introduced, etc.
- **Compartments are not necessarily end2end (while layers above 3 are all end-to-end)**
- **A compartment might include semantics and functions of more than one layer. In fact it may provide the functionality of a complete network stack, or simply the functionality of a single (sub)layer; e.g.:**
  - a compartment may be composed of several/different layers/implementations (TCP over IPv4 and IPv6),
- **A compartment does not have to spawn functionally in all data and control planes (maybe minimally the control plane is sufficient); e.g.:**
  - a compartment may be an overlay network dedicated to some task (e.g. monitoring)
- **Compartments may have more "fluid" (generic) semantics (depending on the mission and the number of planes they spawn across) and are less opaque than layers are in the OSI model**



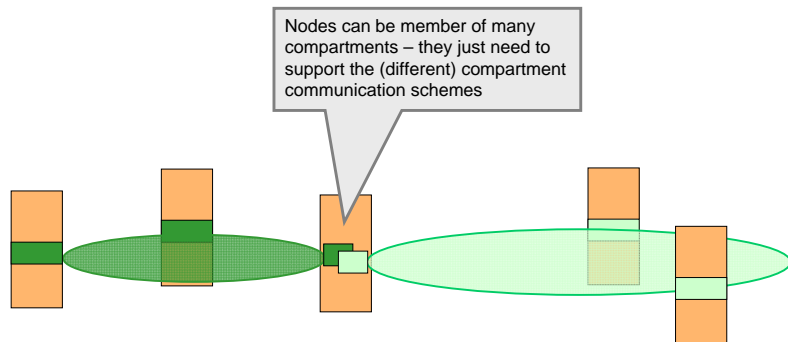
# Intra-Compartment Comm.



**Invariant:** The compartment communication scheme must be supported by all members that want to directly communicate with each other



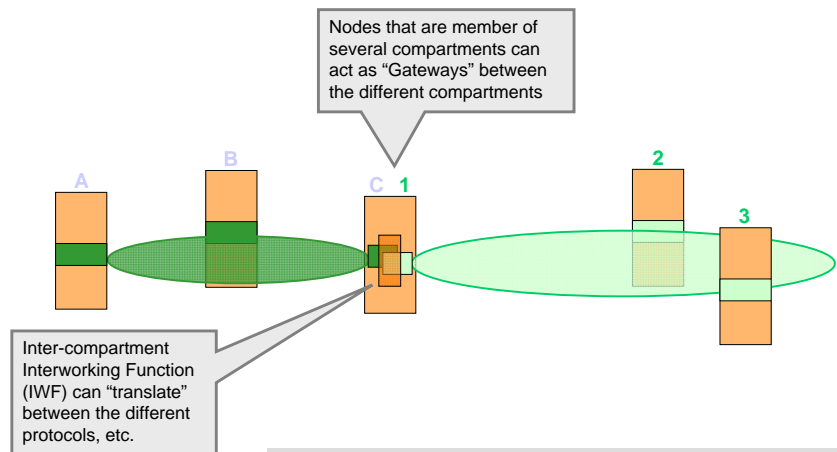
# Multi-Homing



**Limitation:** No inter-compartment communication possible due to lack of common identifier space and the necessary interworking functionality



# Inter-Compartment Comm. between adjacent compartments

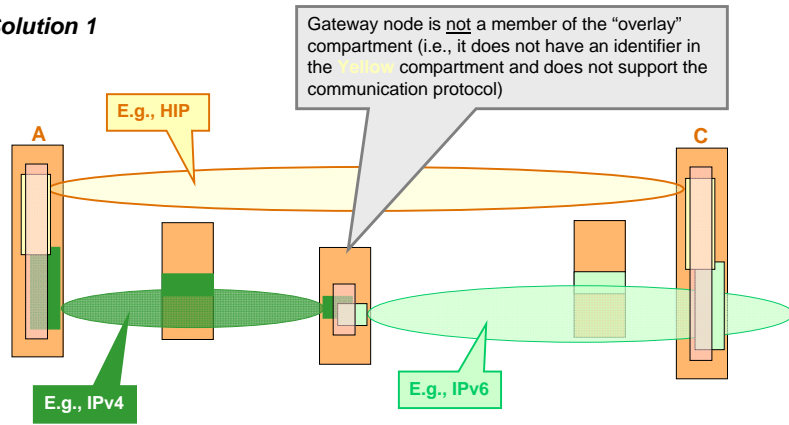


**Limitation:** Despite the availability of end-to-end connectivity, end-to-end communication (apart from broadcast) across compartment boundaries is not possible due to the lack of a common identifier space.

## Inter-Compartment Comm. based on "Overlay" Compartment (I)



### Solution 1

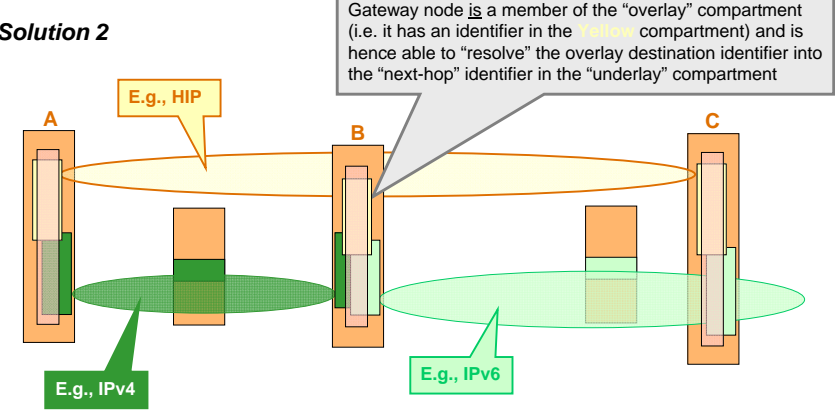


**Limitation:** End-to-end communication between A and C is only possible if the Green compartments "broadcast" all packets to all potential gateways, which then perform the protocol translation. (Note: Only static mapping between identifier spaces is possible)

## Inter-Compartment Comm. based on "Overlay" Compartment (II)



### Solution 2

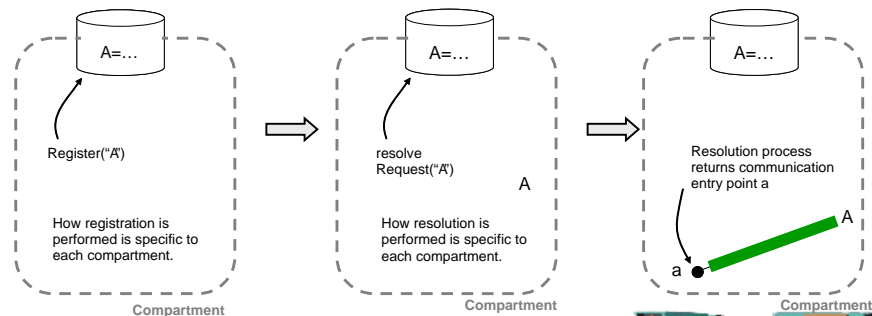


**Note:** This solution also includes the case where the GW node B does not support the actual data/user plane protocol that is used in the Yellow compartment, but just the signalling/control plane protocol, which is used to setup the translation state between the Dark Green and the Green compartment

## Compartment Functionality



- **Registration and resolution are key functionalities of compartments.**
  - Each compartment defines a conceptual membership database.
  - Registration: explicit joining and exposing is required ("default-off" model).
  - Resolution: explicit request before sending ("no sending in the void").



## Naming and Addressing



- **Addressing and naming are left to compartments**
  - Each compartment is free to use any addressing and naming schemes
  - (or is free to not use addresses, for example in sensor networks)
- **The main advantages are:**
  - No need to manage a unique global addressing scheme
  - No need to impose a unique way to resolve names
  - ANA is open to future addressing and naming schemes
- **The main drawbacks are:**
  - Global routing becomes something similar to searching (if communicating parties are not all members of a given compartment)

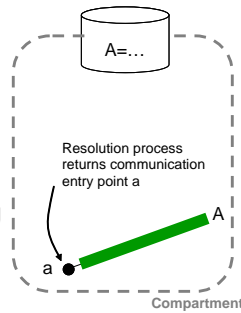
## Local Labels for Handling (global) Addresses



- **"Resolution of members" results in a local label**
  - Addresses (if any) and names (if any) limited as input for resolution
  - Applications send data to labels (which stands for a communication entry point)

- **Properties of local labels:**

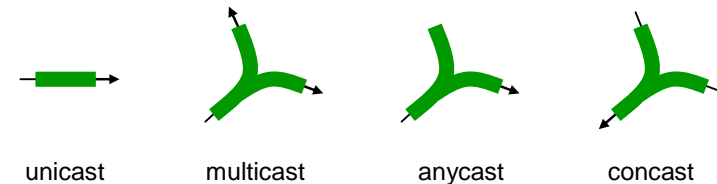
- Size of labels can change from device to device
- Labels' lifetime = communication lifetime (like sockets)
- No need to manage a unique global addressing scheme
- ANA is open to future addressing and naming schemes (via resolution)



## Information Channels (ICs)



- **Resolution process returns access to an "information channel" that can be used to reach the target member(s).**
  - Various types of information channels.

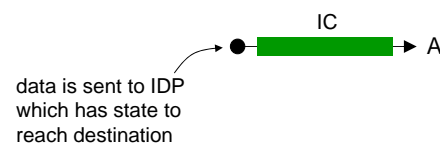


## Information Dispatch Points (IDPs)



- **Startpoints instead of endpoints**

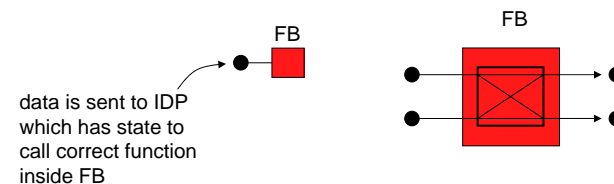
- Communication is always towards a startpoint, or information dispatch point (IDP)
  - Ability to bind to destinations in an address agnostic way.
  - This is important to support many flavors of compartments that can use different types of addresses and names.
  - Useful decoupling between identifiers and means to address them.



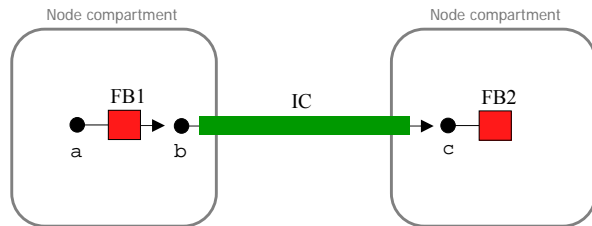
## Functional Blocks (FBs)



- **Code and state that can process data packets.**
  - Protocols and algorithms are represented as FBs.
  - Access to FBs is also via information dispatch points (IDPs).
  - FBs can have multiple input and output IDPs.
  - FB internally selects output IDP(s) to which data is sent.



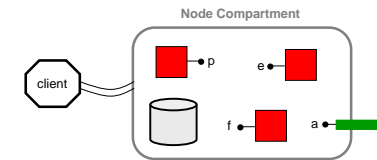
# How ICs, FBs, and IDPs fit together



# Modeling Nodes as Compartments

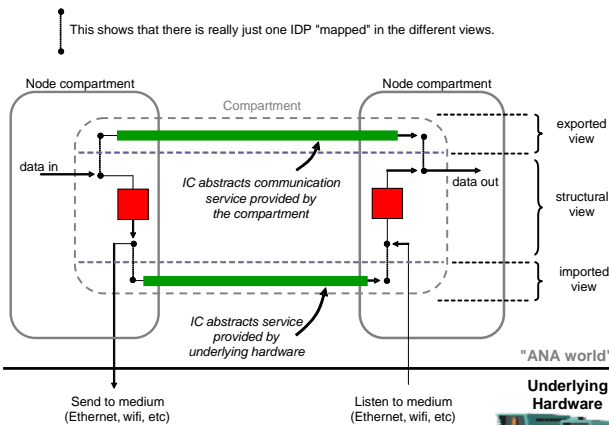
- Organise a node's functionalities as (compartment) members:
  - Member database: catalog of available functions
  - Resolution step to access a given function
    - Also implements access control.
  - Resolution instantiates functional blocks (FBs)
  - The node compartment hosts/executes FBs and IDPs
- Applications first attach to the node compartment:
 

The node compartment is the "startpoint" of any communication.



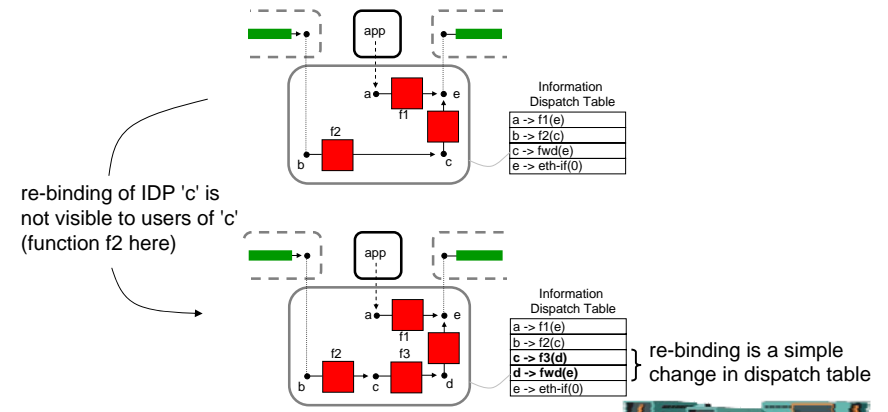
# Different "Views" of a Compartment

- A network compartment has different views, for different usage.



# Functional Composition (I)

- "Chains" of functions are setup on-demand in a dynamic way.
  - Packet dispatching in ANA is based on IDPs.



## Functional Composition (II)



- **Motivation**

- Varying roles of network nodes
- Changing network conditions
- Varying end-to-end paths
- Different application requirements

- **Idea**

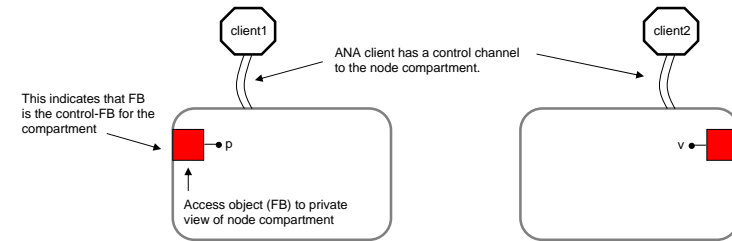
- Customisation of communication structures
  - On-demand creation and removal of custom communication structures



## Example of Communication Setup



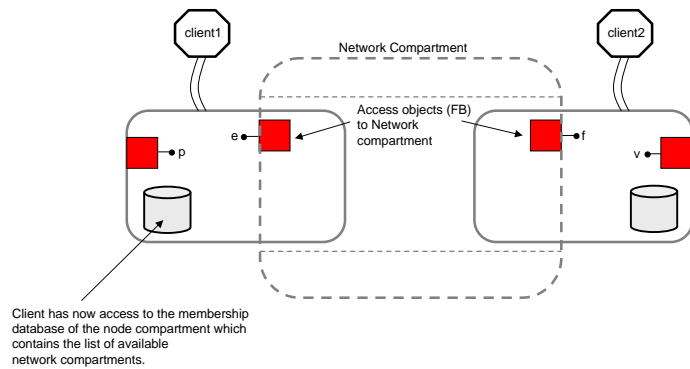
- **Interaction with the node compartment is via a special kind of FB called an "access object (AO)".**
  - For example, register and resolve requests are sent to the AO.



## Example of Communication Setup (II)



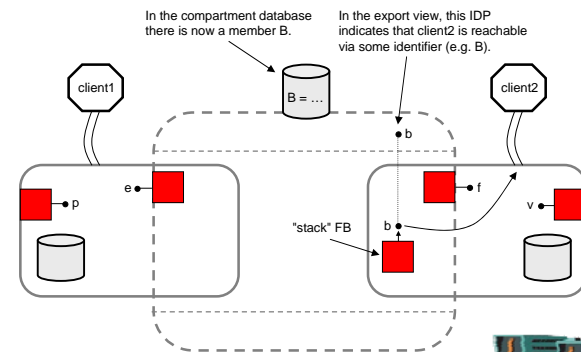
- **Clients get access to the network compartment access objects.**



## Example of Communication Setup (III)

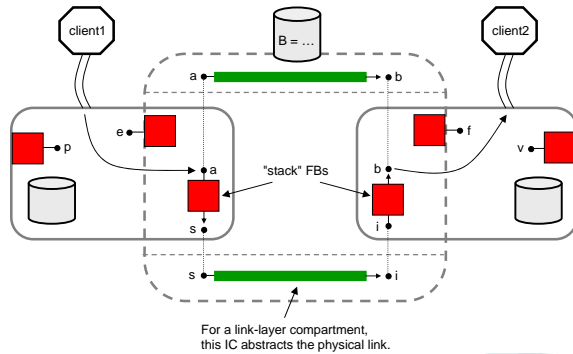


- **Client2 registers (via the IDP 'f') an identifier "B" with network compartment.**
  - Conceptually, this creates an entry in the membership database.



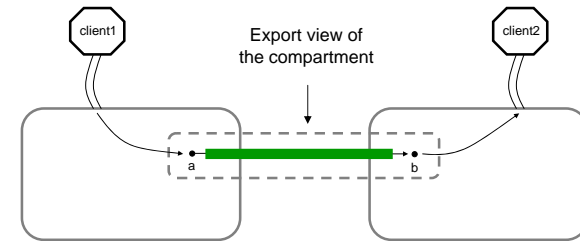
## Example of Communication Setup (IV)

- Client1 resolves (via the IDP 'e') the identifier "B" and receives startpoint IDP 'a'.

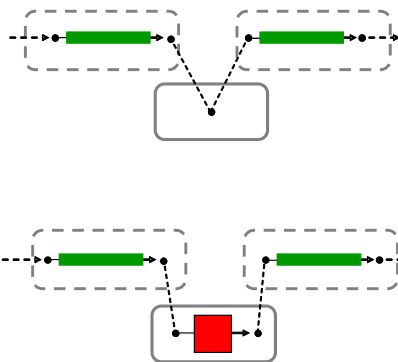


## Example of Communication Setup (V)

- Typically, client1 only sees exported view (unless compartment exposes internal operation).



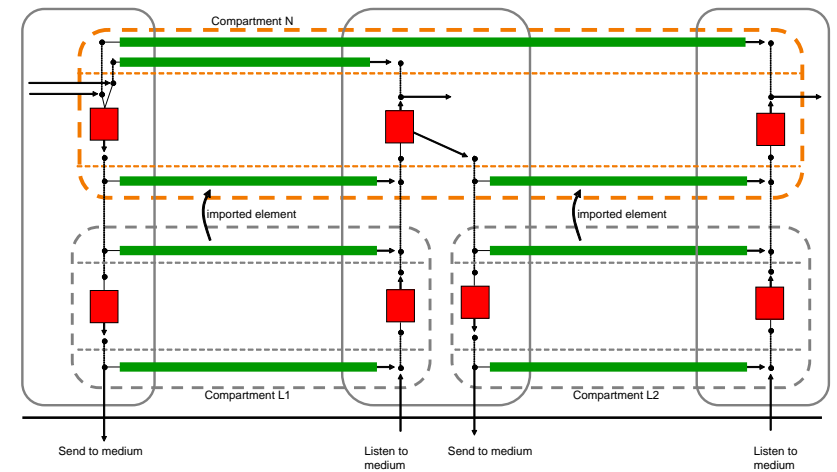
## Forwarding ... Some Examples.



Bridging

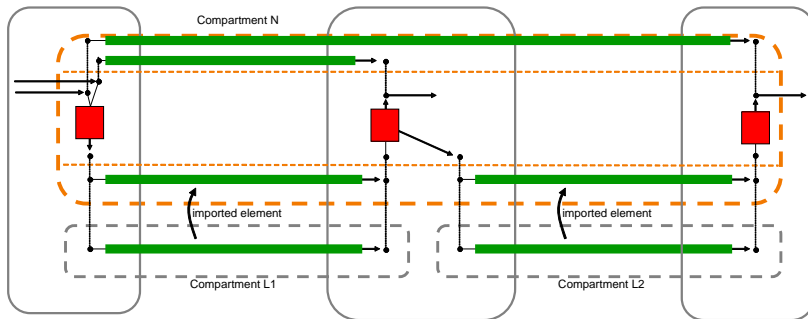
+ intermediate processing

## Overlay Scenario with Compartments



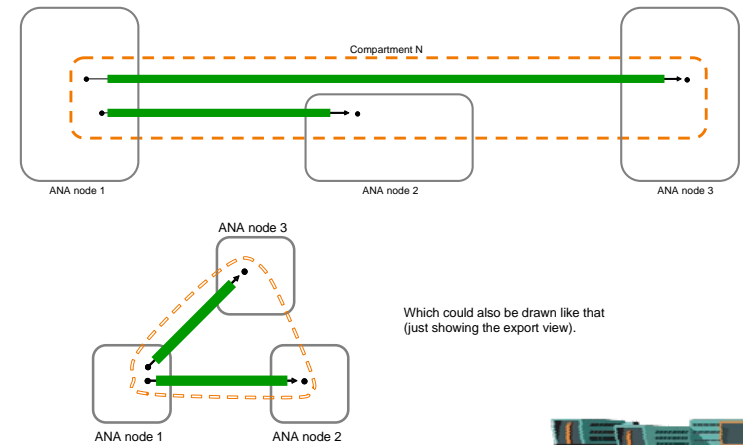
## Overlay Scenario with Compartments (II)

- Same figure but only with exported views of L\* compartments



## Overlay Scenario with Compartments (III)

- Figure just showing export view of compartment N.



Which could also be drawn like that (just showing the export view).

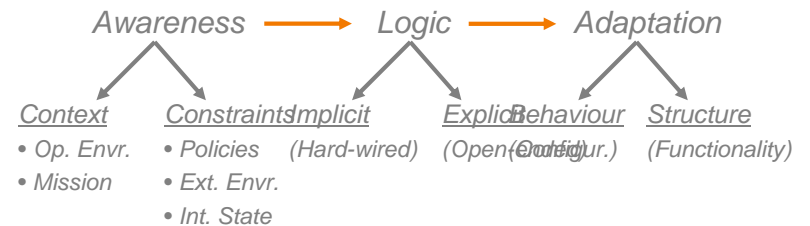
## Functional Composition: Objectives

- A flexible network subsystem that leverages autonomic behaviour
- Provide the machinery for adaptation
- Capture and engineer incentives for adaptation
- Simplified Model

Awareness → Logic → Adaptation

## FC Model - Paradigms

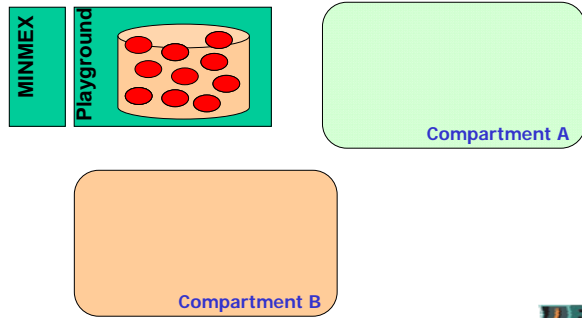
- Model in more detail



- Paradigms (from natural world)

- Instinctive adaptation (mostly behaviour)
- Cognitive adaptation (behaviour/ structure)
- Evolutionary adaptation (structure)

# FCA: The Challenge



Computing Department

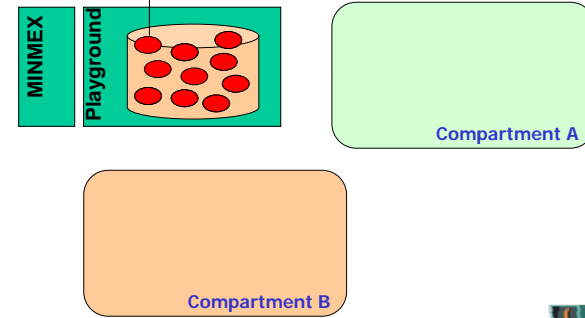


InfoLab21

# FCA: The Challenge



An ANA Node hosts a number of FBs



Computing Department

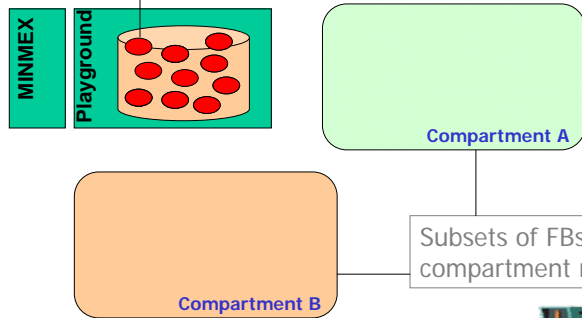


InfoLab21

# FCA: The Challenge



An ANA Node hosts a number of FBs



Computing Department

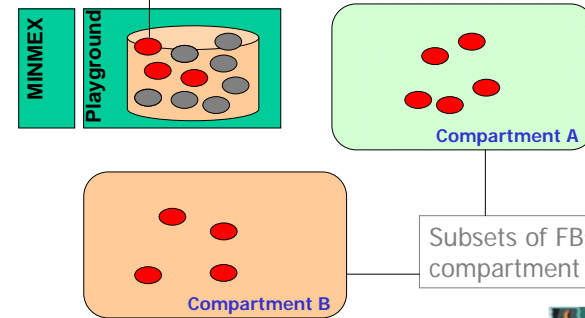


InfoLab21

# FCA: The Challenge



An ANA Node hosts a number of FBs



Computing Department

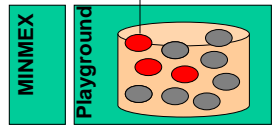


InfoLab21

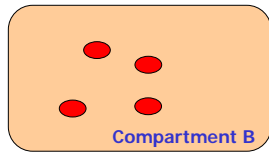
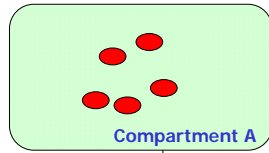
# FCA: The Challenge



An ANA Node hosts a number of FBs



Some FBs provide functionality to more than one compartments (they are agnostic to it)



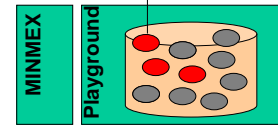
Subsets of FBs serve compartment needs



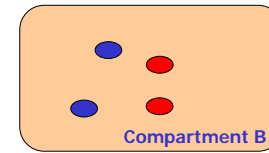
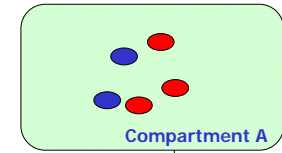
# FCA: The Challenge



An ANA Node hosts a number of FBs



Some FBs provide functionality to more than one compartments (they are agnostic to it)



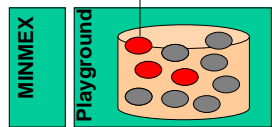
Subsets of FBs serve compartment needs



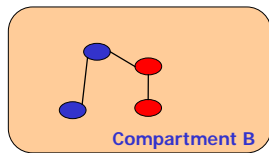
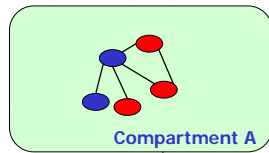
# FCA: The Challenge



An ANA Node hosts a number of FBs



An implicit structure among FBs dictates how they cooperate to provide a compartment's service



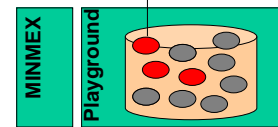
Subsets of FBs serve compartment needs



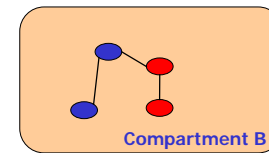
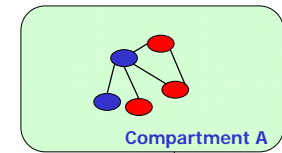
# FCA: The Challenge



An ANA Node hosts a number of FBs



The structure may be static (e.g. legacy Internet) or dynamic over time/space



Subsets of FBs serve compartment needs

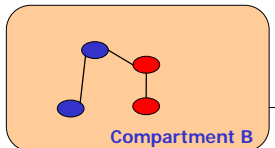
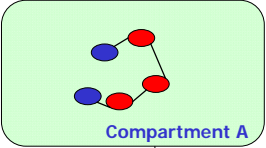
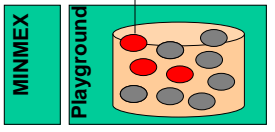


# FCA: The Challenge



The structure may be static (e.g. legacy Internet) or dynamic over time/space

An ANA Node hosts a number of FBs



Subsets of FBs serve compartment needs

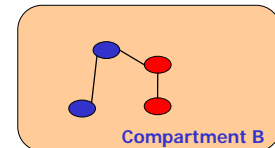
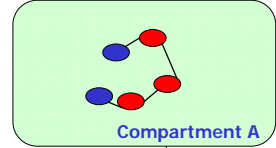
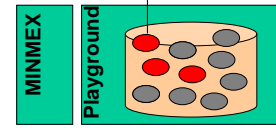


# FCA: The Challenge



Functions may be added or removed in the composite also over time

An ANA Node hosts a number of FBs



Subsets of FBs serve compartment needs

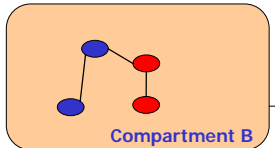
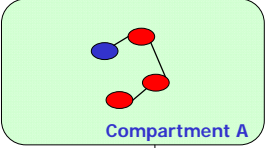
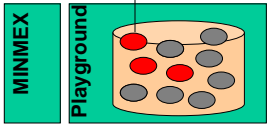


# FCA: The Challenge



Functions may be added or removed in the composite also over time

An ANA Node hosts a number of FBs



Subsets of FBs serve compartment needs

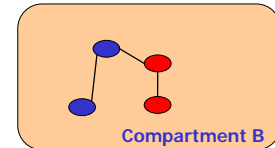
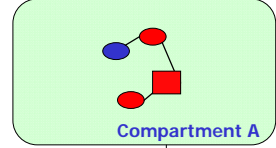
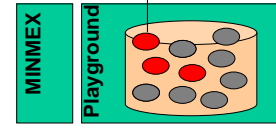


# FCA: The Challenge



Finally an FB may change behaviour over time (re-configuration)

An ANA Node hosts a number of FBs

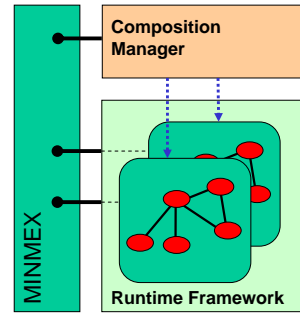


Subsets of FBs serve compartment needs

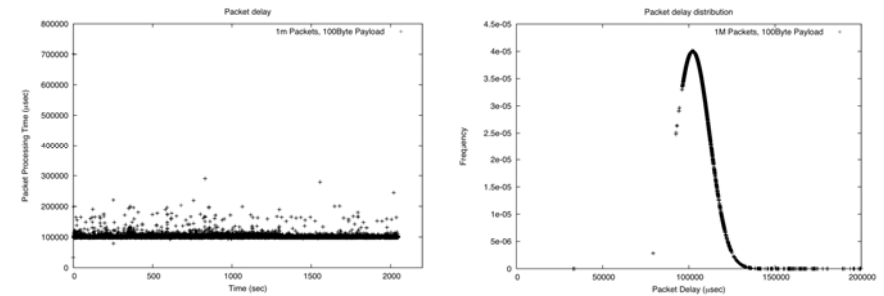


## FCA: Design Outline

- **Designed a system**
  - Provides the expected flexibility and functionality
  - Satisfies the additional requirements
- **Implementation as a set of 2 FBs**
  - Runtime Framework
  - Composition Manager
- **Initial evaluation in user space**
  - Message propagation delay
  - Using a pass-thru classifier



## FCA: Initial Evaluation



21845 messages / run		
Run	Mean (µsec)	Std. Deviation (µsec)
1	105.6	42.2
2	101.6	6.1
3	102.5	9.9

## Objectives

- **Simplified Model**



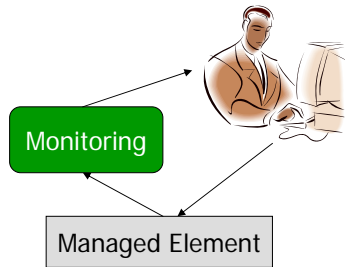
## Monitoring in Autonomic Networks

- **Monitoring Architecture is a service to all functional blocks that need some network state awareness**
  - To base decisions on
  - To protect the system
  - etc.
- **Goals:**
  - Avoid duplication of monitoring tasks at many levels of the architecture (typically in many overlays)
  - Provide resilient and flexible means to store and give access to monitored data
  - Provide means to adapt monitoring to resource availability and to reduce active probing load

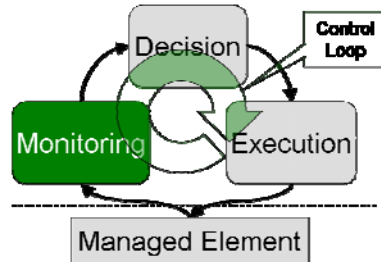


## Monitoring: Classic vs. ANA (I)

### Classic approach



### Autonomic approach



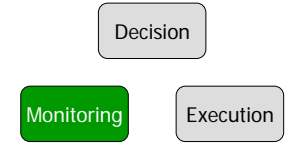
#### Examples of decisions:

- Compose functional blocks differently
- Move service or data elsewhere
- Change routing

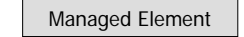
## Monitoring: State-of-the-Art vs. ANA (II)

### Existing monitoring solutions:

- Monitoring is add-on
- Managed element exists
- Need to control
- Design and program monitoring element

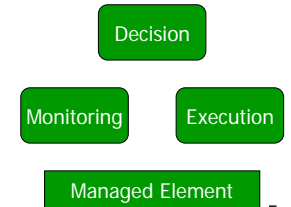


### ANA integrates monitoring as 1st class citizen!



### What is known about managed element, decision, execution at the architectural level?

Nothing!



## Fundamental Challenges

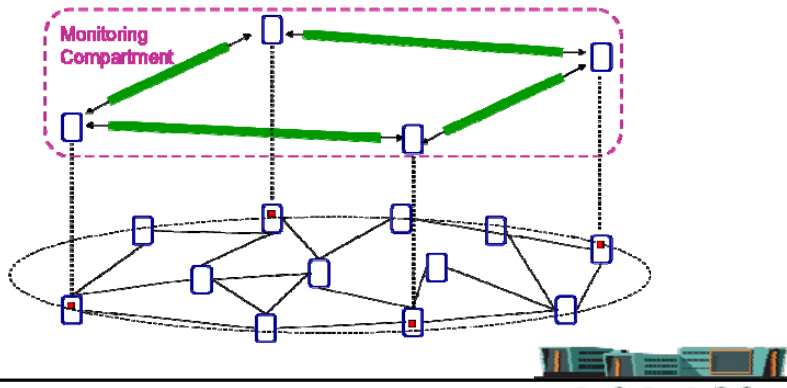
- **No a priori knowledge in the architecture**
  - Network topology
  - Node support, interfaces
  - Tasks
  - Invocation time, execution lifetime
- **Enable monitoring components to explore the available services, interfaces, data types, etc.**
- **Information and Knowledge Management**
- **Dynamic and programmable monitoring**

## Monitoring Concepts

- **Dynamic, adaptive, and programmable monitoring**
  - Trigger monitoring component, e.g. start, pause, resume, terminate
  - Dynamic placement of monitoring components
  - Adapt to changes, e.g. new anomalies to be detected
  - Specify the monitoring tasks at run-time, e.g. CQL for Data Stream Management Systems

- **Monitoring compartments with self-\* properties**

- Monitoring at one location is often not sufficient
- Distributed and cooperative monitoring

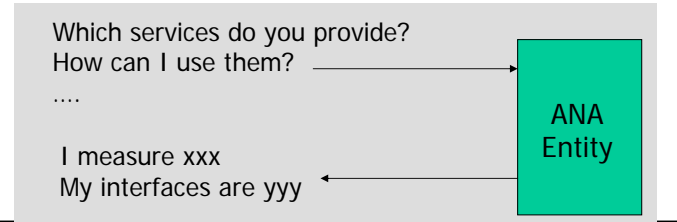


- **No a priori knowledge**

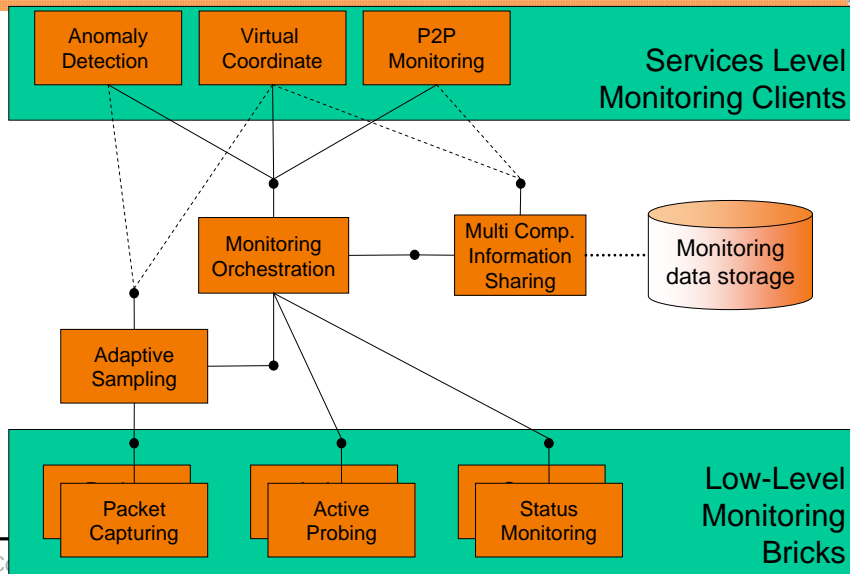
- Maximum flexibility
- Everything needs to be dynamically explored

- **The fundamental ANA standard:**

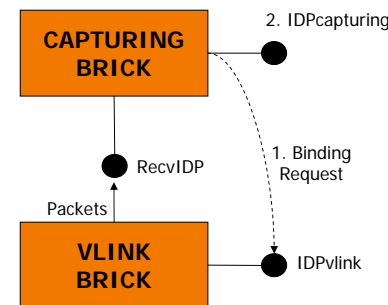
- A unified generic interface
- Self-describing



# Node Architecture for Monitoring

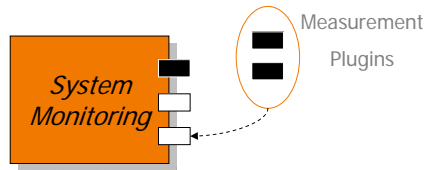


# Capturing Brick

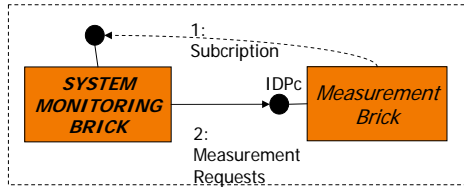


- **Lowest level component**
- **Passively collects packets**
- **Allows multiple consumers**
- **Provides packet headers inspection**
- **Provides configuration capabilities:**
  - Filter rules
  - Capturing state (start, stop, pause)

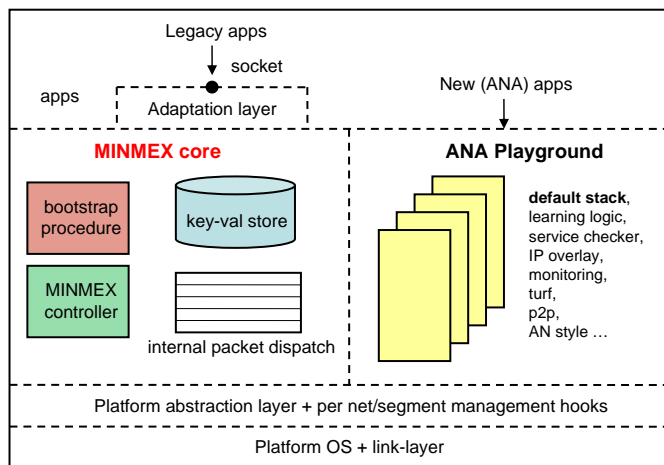
1. **Binding request to VLINK, specifying RecvIDP**
2. **Waits for monitoring requests at IDP capturing**



- Measures critical system parameters
- Manages interactions between consumer bricks and measurements bricks (“plug-ins”)
- Measurement brick subscribes (1) and gives IDP to receive requests
- Provides a multi-mode interface:
  - On-request
  - On-timer
  - On-condition



- The MINMEX (Minimal INfrastructur for Maxmimal EXtensibility)
  - The idea
    - invariant, generic, simple, agnostic, neutral . . .
    - not autonomic, not the full thing, not reflective. . .
    - but must be “compartment aware”
  - MINMEX as “the ANA micro-kernel”
    - Supports core API and inter-brick communications
    - Implements packet dispatching to IDPs
    - Implements the Node Compartment
  - MINMEX as the invariant part of ANA
    - Bare support for function lookup, composition and internal packet routing
    - Contains default primitive network for network bootstrap
    - Provides a variable “playground”
- “Bricks” i.e., individual components of the ANA Playground
  - Can be a functional block which offers access to a network compartment
  - Can also provide processing support (e.g. encryption)
  - Brick eXchange Points (BXP) for inter-Brick interaction



- The ANA node can be distributed
- Not all the components of an “ANA Node” have to run on the same computer
  - The MINMEX and its bricks can communicate via various IPC types called “gates”: Unix/UDP sockets, named pipes, generic netlink
  - The motivation was that the notion of a “node” was not restricted to being a physical device

## Basic Design: API



- Three different API levels.
- **Objective: better understand the level of complexity vs. flexibility we want to reach.**
  - API Level 0: maximum flexibility but developer must know all the details for encoding and decoding messages.
  - API Level 1: good flexibility and developer can use functions to encode and decode messages.
  - API Level 2: less flexibility, but function prototypes are very ease to use, code is easy to write.



## Basic Design: Platforms



- **One code, two platforms**
- **The base code compiles as either userspace application or Linux kernel modules**
  - Userspace: easy for development and debugging, easy to use, most people can use it
  - Linux kernel: for best performance, permits to interact with kernel network internals
- **Bricks can be developed in a "platform-agnostic" way according to a standard template**
  - The API library provides wrapper functions and mechanisms to properly handle function calls (e.g., malloc vs. kcalloc, main vs. init)
  - We also provide "agnostic" libraries for system-specific functions such as threads and timers
  - Passing arguments to bricks via CLI is also done via a system-agnostic mechanism (à la argc/argv[ ])



## Available components



- **MINMEX: node compartment, IDT, IDP manipulation, status interface, API libraries.**
- **Bricks:**
  - Ethernet and IP compartments.
  - vlink sub-system for flexible "cabling" of ANA nodes.
  - *Monitoring components*: core part, packet capture, measurements (CPU, net load), "ping".
  - Inter-compartment routing with regular expressions.
  - Content centered routing, Field Based routing
  - Functional composition prototype.
  - User side: chat application, various code examples.
  - And many other bricks of project partners
- **Tools**
  - QuickRep, timers, threads, Remote IDP Access



## Conclusions



- **Autonomic Networking Architecture**
  - Framework for implementing autonomic principles
  - New communication concepts
    - Functional compositions
    - Compartments
    - Functional blocks
  - The role of monitoring
    - Central enabler of autonomic behaviour
- **Basic Design Principles**
  - MINMEX as micro-kernel to
  - Bricks as basic units for functional blocks



## Acknowledgements



Many thanks to all the colleagues of the ANA project who have contributed to these slides



Computing Department



InfoLab21

## Questions



Computing Department



InfoLab21