The ANA Project: FISS 09

ANA Blueprint

G. Bouabene (UBasel)

Bremen, Germany

July 20, 2009

ana

autonomic

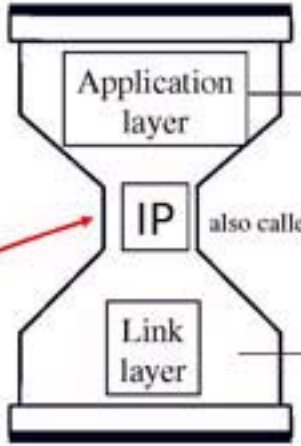
network architecture

A new architecture, why ?

- The Internet suffers from architectural stress:
  - Not ready to manage the envisaged huge numbers of devices
  - New technologies (sensor networks, mobility, personal area networks etc) difficult to integrate
  - Lacks integrated monitoring and security mechanisms
  - Other issues such as quality of service and more.

Motivation

- Variability in the Internet is above and below IP: it's the "hour-glass" model.



www, email, ftp, ssh, DNS,
peer-to-peer (eMule, BitTorrent)
VoIP (Skype), VoD, grid, …

also called the "waist" of the Internet

Ethernet, wifi (802.11), ATM,
SONET/SDH, FrameRelay,
modem, ADSL, Cable, …

Changing/updating the Internet core (e.g., IPv6, Multicast, MIP, QoS, …) is difficult or impossible !

Motivation

- Solutions adopted so far :
  - Patches to cope with challenges contradict the initial design paradigms
    - Incoherences → patches to the patches

Consensus in the research community that a next step beyond the Internet is needed.

Motivation

- Projects and initiatives such as
  - GENI, FIND (USA).
  - FIRE (EU): 4WARD, Onelab2, Trilogy
- In the literature:
  - Plutarch, NewArch (RBA, FARA), Turfnet, Selnet.
  - Ambient Networks.
  - RNA: Recursive Network Architecture at ISI.
  - … and many more (old and new).
- Partial proofs of concept, no full architecture (yet).
  - Selnet, M-FARA, Ambient Networks.

ANA: the least common denominator

Our contribution: enable & demonstrate autonomic networking.

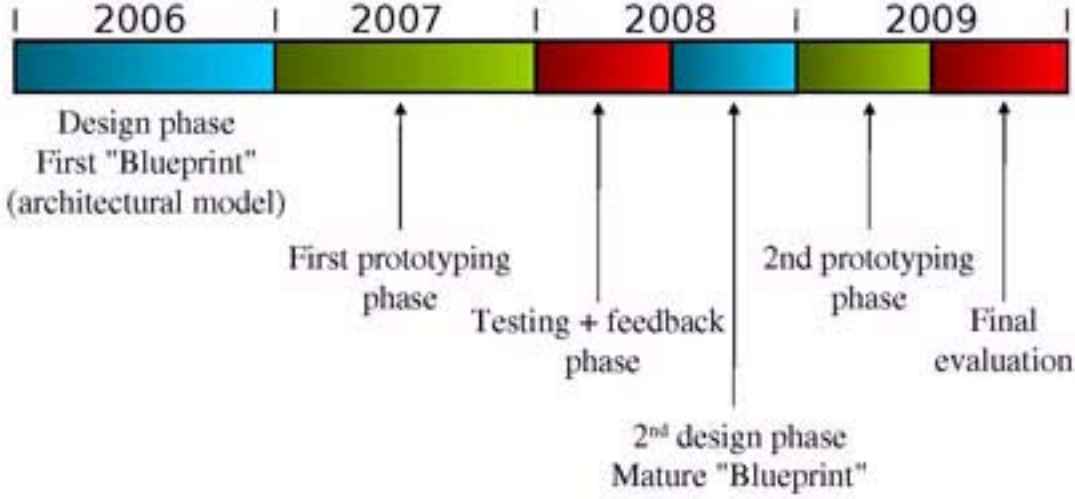
- ANA abstractions allow variability at all levels of the architecture:
  - variants to perform a given task,
  - and networks co-exist and (can) compete, open for extensions (evolution)

Pitfalls in network architecture design

- Static/rigid standards instead of mechanisms for change management
  - Global address space (requires uniqueness and global coordination)
- Leaking of and relying on network internal details
  - Built-in address dependency (i.e. address-centric architecture)
- To avoid running into such pitfalls, we adopt an incremental approach via prototyping cycles
  - Helps revealing faults or black-holes in the architecture design

Growing the architecture

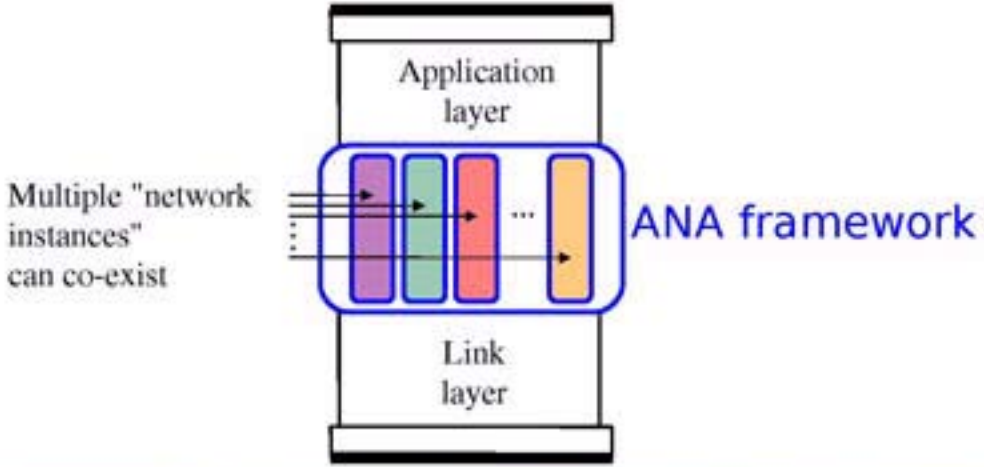
- Project is articulated around 2 prototyping cycles.
  - Methodology: design, test/validate, refine.



ANA Blueprint
a look from inside

ANA ≠ "one-size-fits-all"

- ANA does not want to propose another "one-size-fits-all network waist".
  - ANA is a **meta-architecture** to host, interconnect and federate multiple heterogeneous networks



Core abstractions

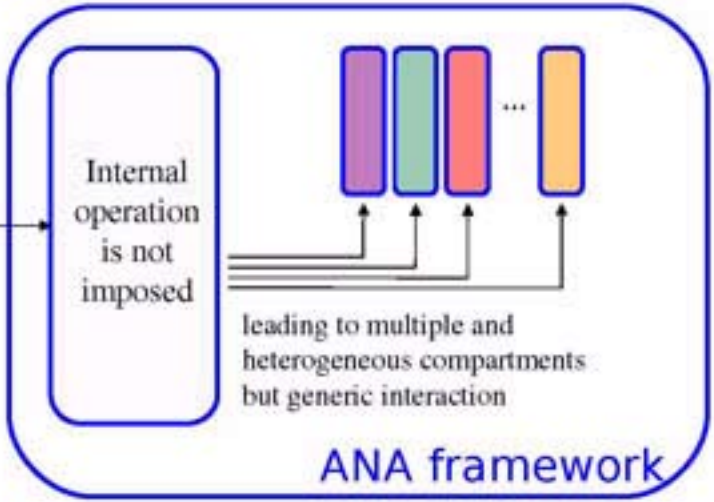
- ANA abstractions:
  - Compartment.
  - Information Channel (IC).
  - Information Dispatch Point (IDP).
  - Functional Block (FB).

- More tricks with compartments:
  - "Node compartment".
  - Overlays and compartment inter-stitching.

ANA Compartment

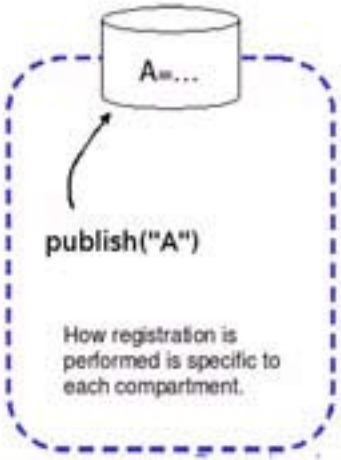
- Compartment = wrapper for networks.
- ANA does not impose how network compartments should work internally: the ANA framework specifies how networks interact.

ANA specifies interfaces and interactions with any network compartment



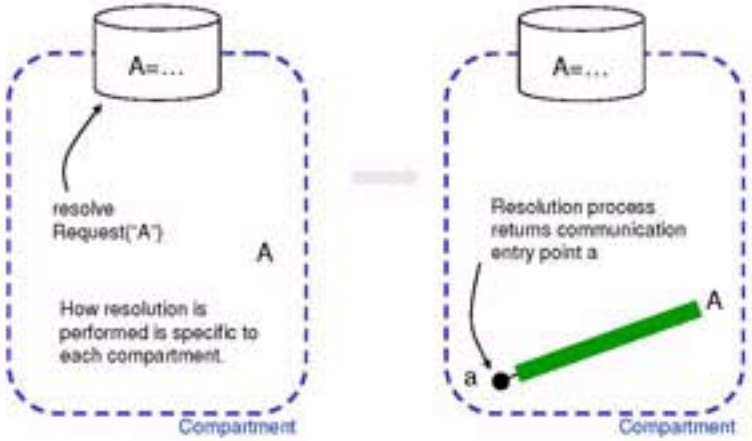
Compartments (cont.)

- A (network) compartment defines how to join and leave a compartment: **member** registration, trust model, authentication, etc.
  - Each compartment defines a conceptual membership database.
  - Registration: explicit joining and exposing is required ("default-off" model).



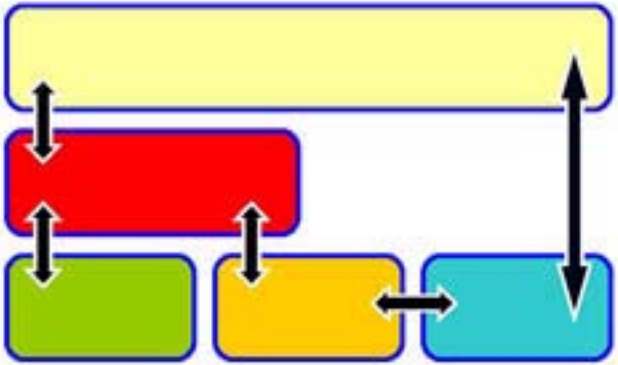
Compartments (cont.)

- Defines **How to reach** (communicate with) another member: peer resolution, addressing, routing, etc.
- Resolution: explicit request before sending ("no sending in the void").



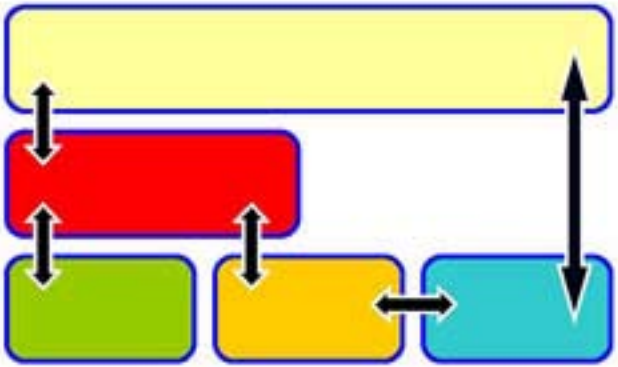
Compartments (cont.)

- Compartments can be overlaid, i.e. compartments can use the communication services of other compartments.
  - compartment abstraction serves as the **unit for the federation** of networks into global-scale communication systems.
  - Has **interaction rules** with "external world", the compartment **boundaries** (administrative or technical), peerings with other compartments, etc.



Compartments (cont.)

Compartments decompose communication systems and networks into smaller and easier manageable units.



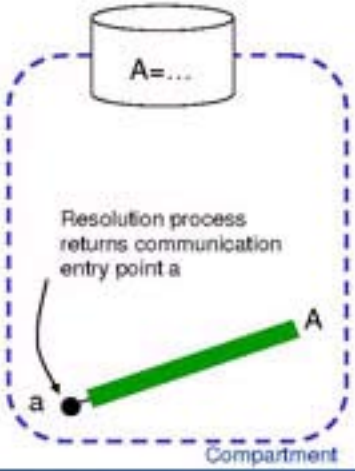
What about addresses and names ?

- 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/challenges are:
  - Back to the CATENET challenges : How to inter-work in such heterogeneous address/name spaces ?

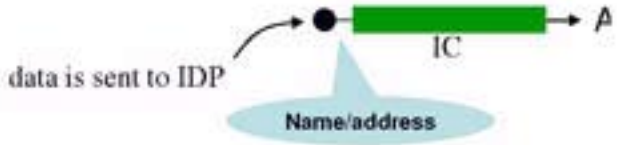
Local labels for addresses

- Target resolution returns a local label = IDP
  - Addresses (if any) and names (if any) limited as input for resolution
  - The IDP maintains the state to reach the destination



Local labels for addresses

- Applications send data only to IDPs
  - Bound to a flexibe element



Startpoints instead of endpoints.

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.

Core abstractions

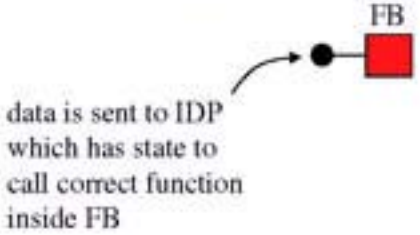
- ANA abstractions:
  - Compartment. ✓
  - Information Channel (IC). ✓
  - Information Dispatch Point (IDP). ✓
  - Functional Block (FB).

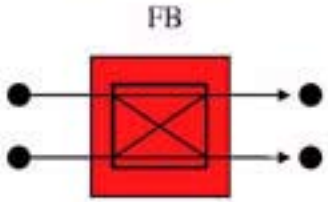
- More tricks with compartments:
  - "Node compartment".
  - Overlays and compartment inter-stitching.

Functional Blocks (FBs)

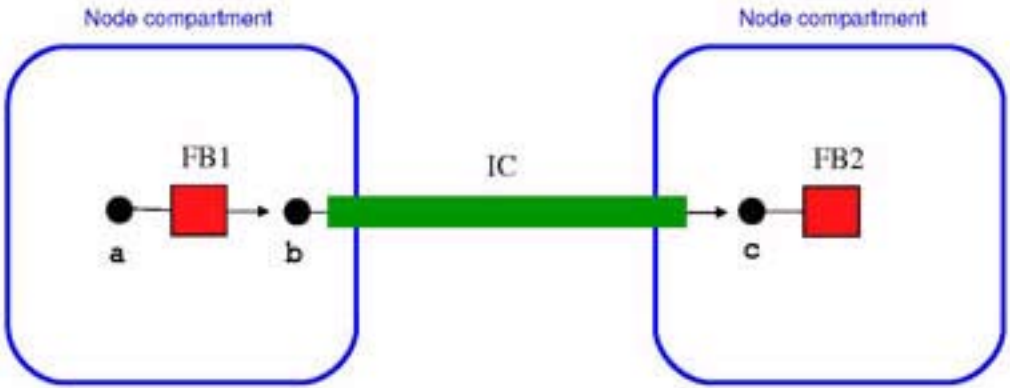
Code and state that can process data packets.

- Protocols and algorithms are represented as FBs.
  - FBs implement the Information Channels (abstract entities)
  - 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.



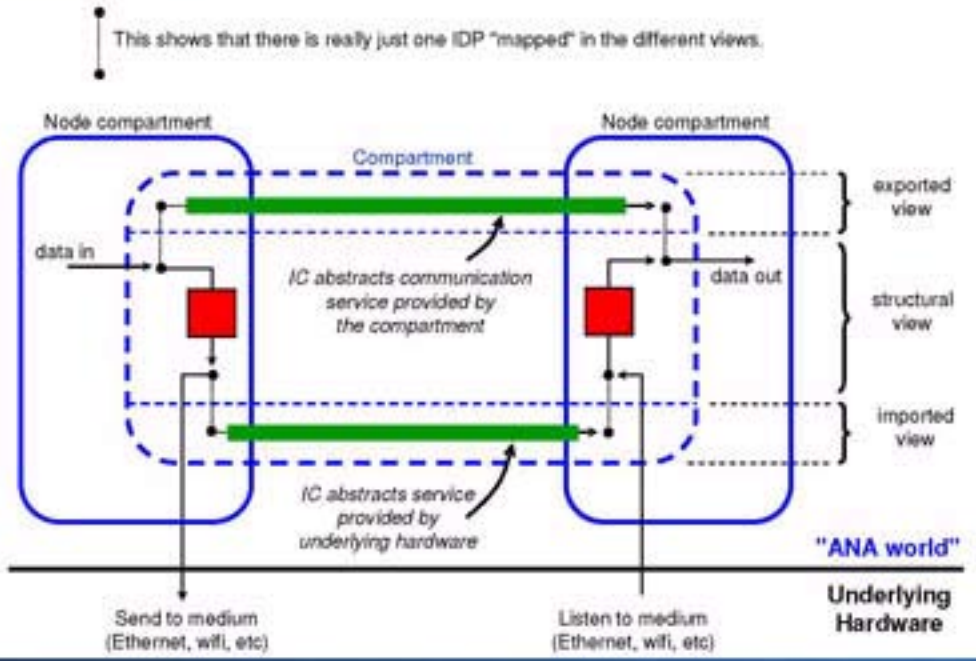


How ICs, FBs, and IDPs fit together



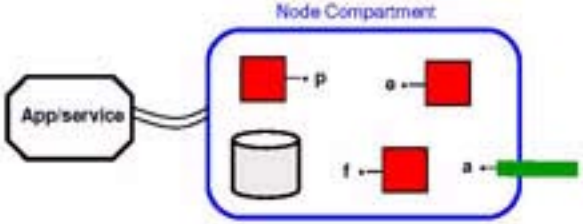
Different "views" for a compartment

A network compartment has different views, for different usage.

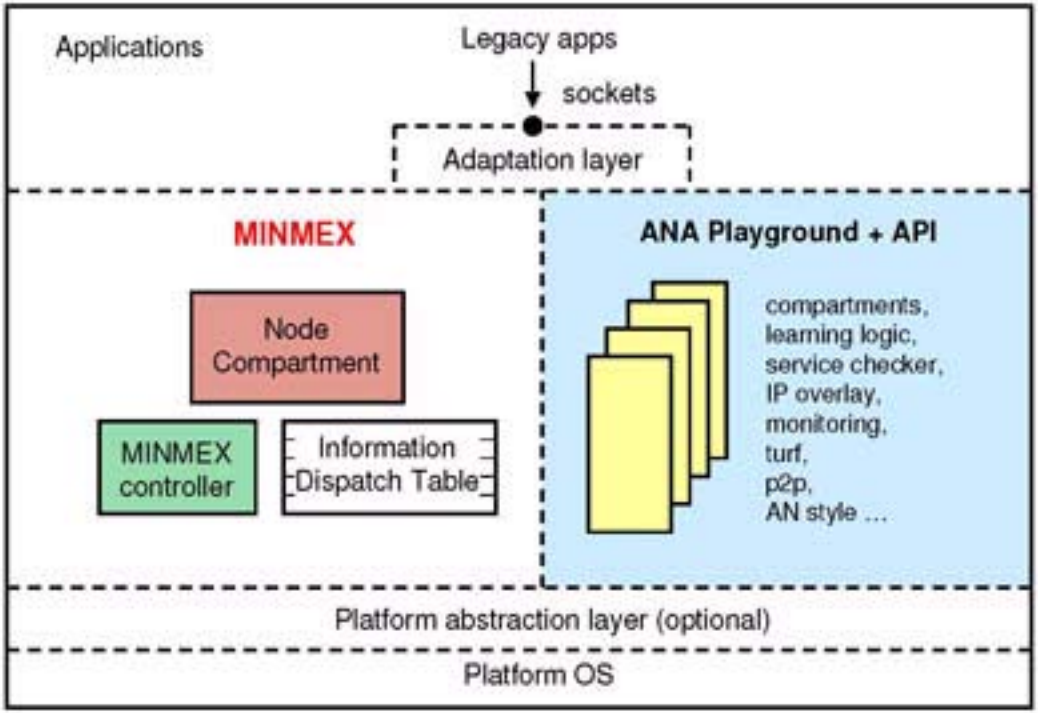


Modelling nodes as compartments

- Organize 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.



Implementation preview



Related publications

- Ariane Keller, Theus Hossmann, Martin May, Ghazi Bouabene, Christophe Jelger, and Christian Tschudin, **A System Architecture for Evolving Protocol Stacks**, *in Proceedings of the 17th International Conference on Computer Communications and Networks (ICCCN'08)*, August 2008, St. Thomas, USA.
- Ghazi Bouabene, Christophe Jelger, Christian Tschudin, Stefan Schmid, Ariane Keller, Martin May, **The Autonomic Network Architecture (ANA),** *submitted to JSAC special issue on Recent Advances in Autonomic Communications.*
- **ANA Blueprint: version 2.0**

Questions ?

Thank you for your attention.

The ANA Project: FISS 09

ANA Communications API

G. Bouabene (UBasel)

Bremen, Germany

July 20, 2009

ana

autonomic

network architecture

Motivation

- Network compartments are free to internally run whatever addressing/naming schemes, routing protocols, etc.

- The "glue" for all interactions in ANA is the **compartment API**.

- All network compartments must support the API in order to allow all possible interactions between compartments.

Overview

- The API offers 6 fundamental primitives.

IDP_p publish (IDP_c, CONTEXT, SERVICE)

int unpublish (IDP_c, IDP_p, CONTEXT, SERVICE)

IDP_r resolve (IDP_c, CONTEXT, SERVICE, chanType)

int release (IDP_c, IDP_r)

void* lookup (IDP_c, CONTEXT, SERVICE)

int send (IDP_r, DATA)

CONTEXTS and SERVICES

- The SERVICE argument is typically ***what*** is being published or looked up.
  - e.g., an address, a name, a file, a video stream, a printing service, etc.

- The CONTEXT defines some ***scope*** inside a compartment.
  - IPv4: 1.2.3.4, 224.0.0.1, 10.1.2.255.
  - IPv6: 2001::1, FF02::1, ::1.
  - eMule: Madonna, Pink Floyd, Blade Runner.

CONTEXTS and SERVICES

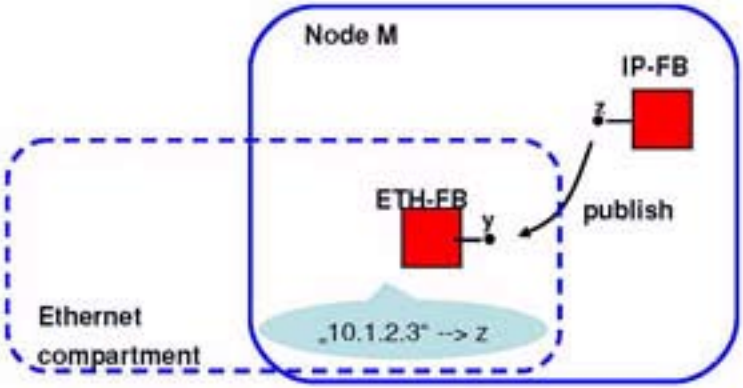
- We have currently specified two "well-known" CONTEXT value.
  - "." → node-local
  - "*" → largest possible scope as interpreted by the compartment

Examples:

- IPv4 compartment:
  "*" ~ 255.255.255.255 "." ~ 127.0.0.1
- Ethernet:
  "*" ~ FF:FF:FF:FF:FF:FF "." ~ local address

Using the API: some examples

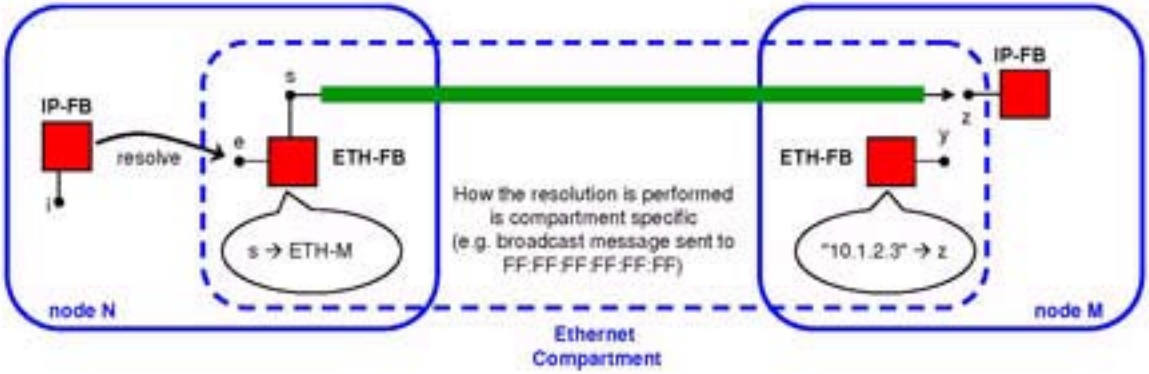
Publishing an IPv4 address in the Ethernet compartment.



z <-- publish(y, "*", "10.1.2.3")

Using the API: some examples

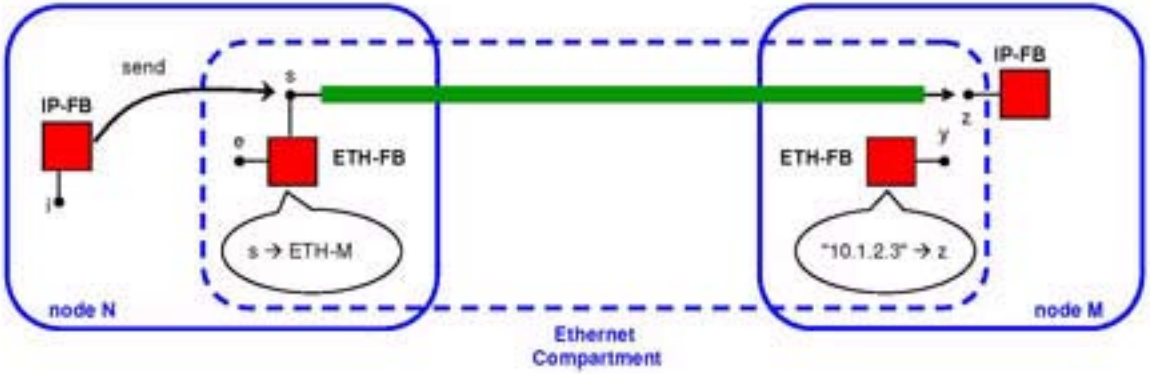
Resolving an IPv4 address in the Ethernet compartment.



s <-- resolve(e, "*", "10.1.2.3")

Using the API: some examples

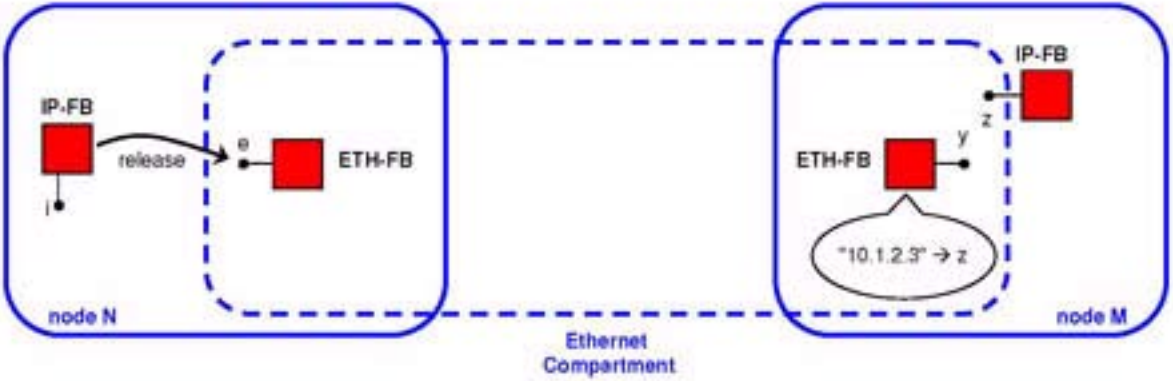
Sending data.



send(s, DATA)

Using the API: some examples

Releasing an information channel.



release(e, s)

Using the API: some examples

The lookup primitive.

List of hosts <-- lookup(d, *, “Chat”)

with IDP 'd' bound to the Ethernet compartment.

List of songs <-- lookup(r, "Rock", “Hendrix”)

with IDP 'r' bound to some online Juke-Box compartment.

Next steps

- The lookup primitive requires further work in order to better define its role and the format of the data it returns.

- e.g. Lookup can be seen as a way to extract inner-network infomation :

List of next hops ← lookup(d, "1.2.3.4", "TCP")

with IDP 'd' bound to the IP compartment.

Related publications

The API was presented in the following papers:

- G. Bouabene, C. Jelger, and C. Tschudin, **Virtual Network Stacks**, *SIGCOMM PRESTO Workshop*, Seattle, USA, August 2008.

- Ariane Keller, Theus Hossmann, Martin May, Ghazi Bouabene, Christophe Jelger, and Christian Tschudin, **A System Architecture for Evolving Protocol Stacks**, *in Proceedings of the 17th International Conference on Computer Communications and Networks (ICCCN'08)*, August 2008, St. Thomas, USA.

Questions ?

Thank you for your attention.

The ANA Project: FISS 09

ANA Core implementation

G. Bouabene (UBasel)

Bremen, Germany

July 20, 2009

ana

autonomic

network architecture

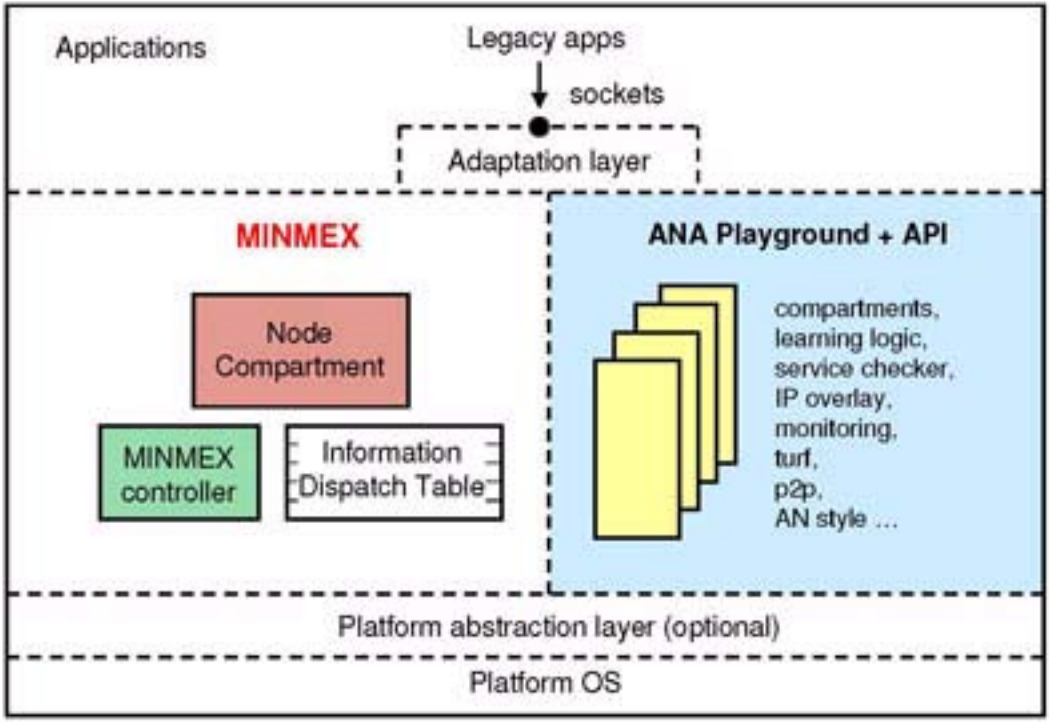
Software releases

- First public release of software done on July 2008.
- Release of the second prototype done in February 2009.

Development process

- Prototyping in ANA is a key research instrument.
  - "Learn by doing", "Grow the architecture".
  - Rationale: a constant prototyping effort is part of the research
    - Helps validate good established concepts
    - Identifies faulty architectural concepts
    - Points out concepts left behind

Basic design



Basic design

Two main components:

- The MINMEX i.e., the ANA "micro-kernel"
  - Supports core API and inter-brick communications.
  - Implements packet dispatching to IDPs.
  - Implements the *Node Compartment*.

- "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).

Basic design

The ANA node can be distributed.

- We do not enforce that all the components of an "ANA Node" 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

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

One code, multiple 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.
- The code has been ported to MacOSX and portable devices : iPhone, Android, Nokia80x
- We also started a support for ANA bricks written in the Erlang Programming language

Basic design

One code, multiple platforms.

- 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. kmalloc, 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.

Available components

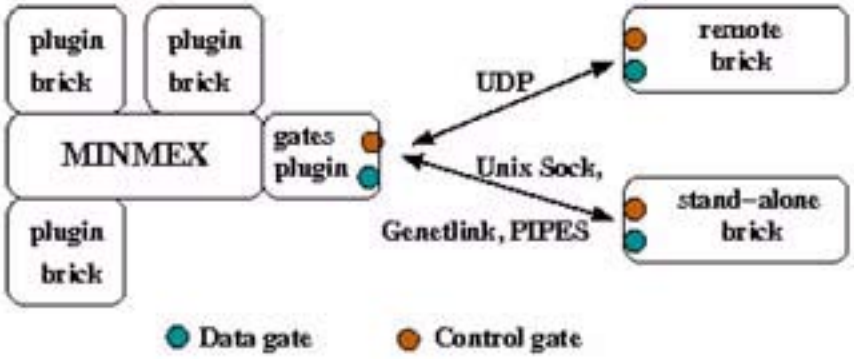
- Bricks:
  - Content centered routing, Field Based routing
  - Functional composition prototype.
  - User side: chat application, various code examples.
  - And many other bricks of project partners
- Tools
  - QuickRep, timers, threads, Remote IDP Access

Plugin model

- Plugin model for the Minmex :
  - Allows for easy extension of minmex functionality
  - .so plugins for user-space
  - .ko plugins for kernel
  - Allows to attach bricks directly to the minmex
    - Direct function calls → good performance
  - IPC handling at the minmex in a separate plugin
    - Releaves the minmex from some code complexity

Plugin model

Plugin model for the Minmex :



Features

- Easy build system :
  - Operates via a front-end control file
  - Allows users to indicate which bricks to compile and in which mode (user-space or kernel)
- Template makefiles provided to ease brick's integration

Features

The prototype also incorporates novel architectural concepts :

- IDP information repository :
  - Allows to store and retrieve information regarding IDPs.
  - The information contains :
    - The view to which the IDP belongs
    - In case attached to a local brick : the next hop IDP
    - In case to an IC : the CONTEXT and SERVICE of the destination and the MTU of the channel

Features

The prototype also incorporates novel architectural concepts :

- Event Notification system :
- Allows bricks to keep alert regarding events in the node compartment
  - Works in a pub-sub fashion
  - The events notify about :
    - Deletion, redirection, IDP busy/down, info change of IDPs
    - Attachment, Detachment of bricks to the minmex

Questions ?

Thank you for your attention.