Policy 2007
13-15 June 2007 - Bologna, Italy

XACML-Based Composition Policies for Ambient Networks

Carlos Kamienski (cak@ufabc.edu.br)

Joseane Fidalgo (joseane@gprt.ufpe.br)

Ramide Dantas (ramide@gprt.ufpe.br)

Djamel Sadok (jamel@cgprt.ufpe.br)

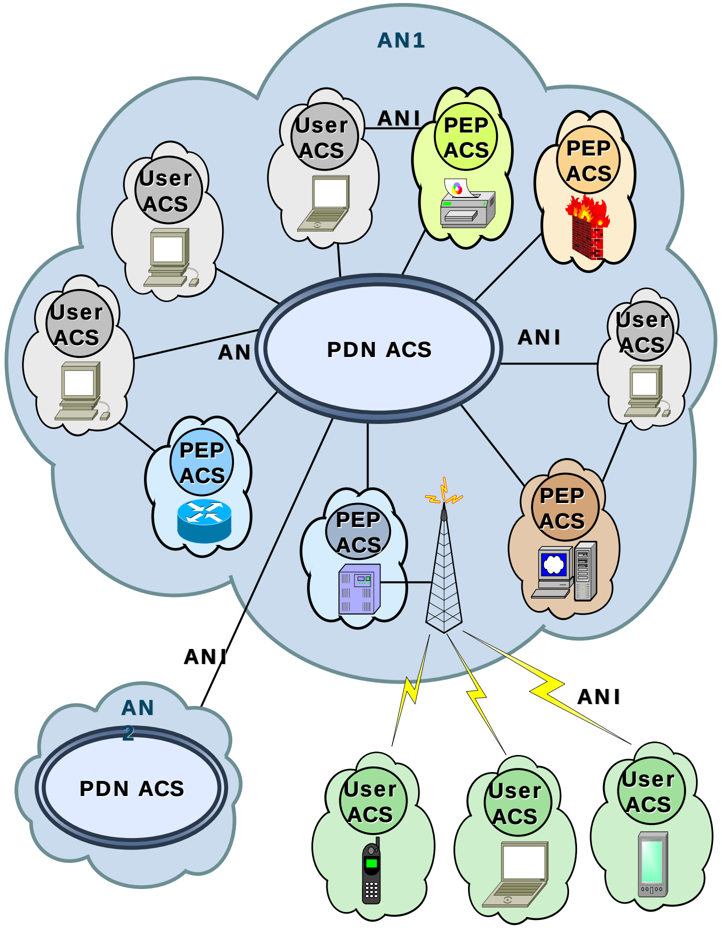
Börje Ohlman (Borje.Ohlman@ericsson.com)

Introduction

- Ambient Networks (AN): new challenges to the management discipline
  - The key concept is **network composition**, for allowing instant and dynamic access to services and resources
- Policies: adequate solution for providing
  - Flexibility
  - Distributed control
  - Self-management
- Traditional management approaches not designed to deal with Internet services for mobile users

Previous Experience

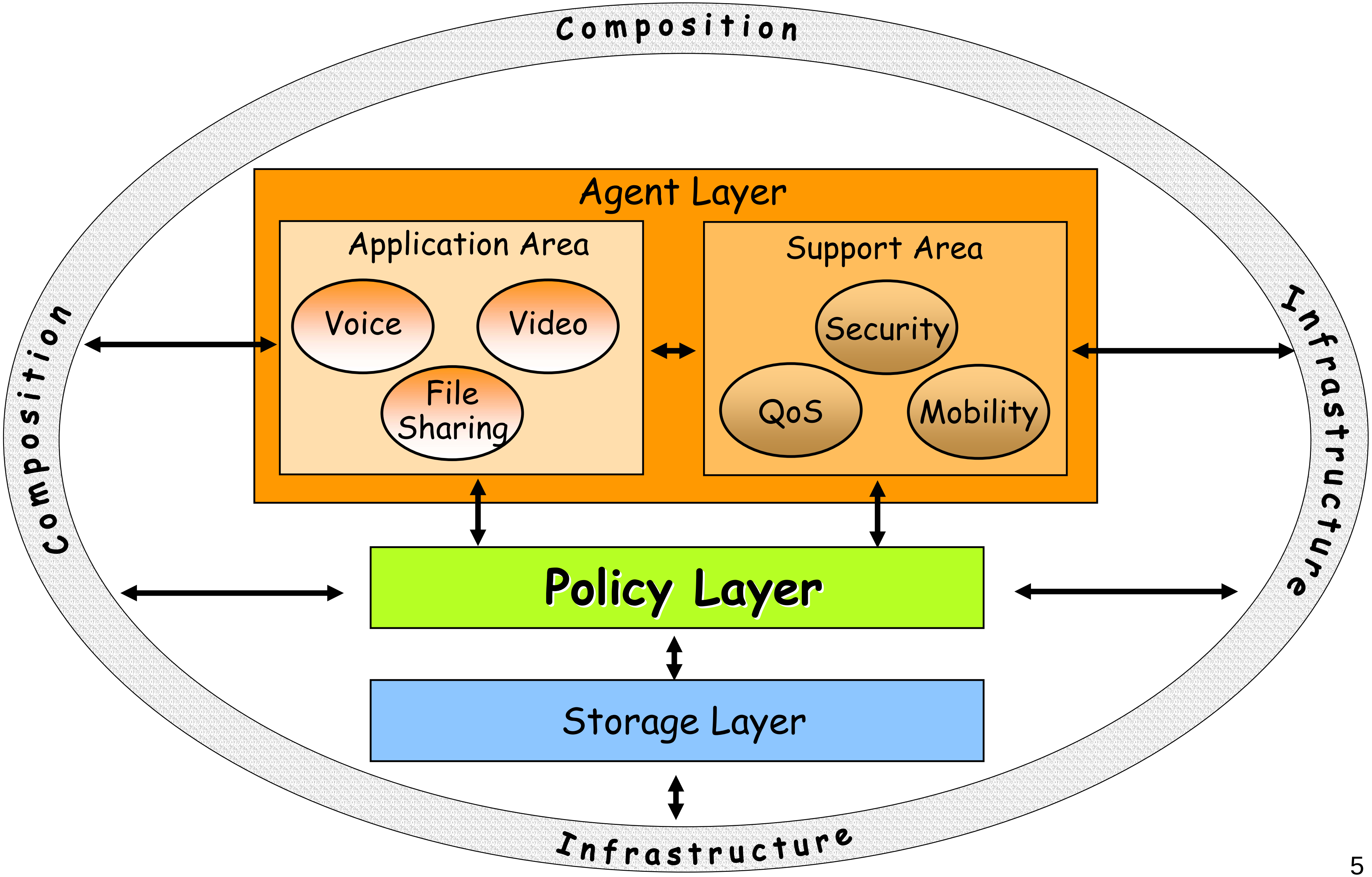
- Design and implementation of a P2P-based version of PBMAN
  - PBMAN = Policy-based Management Framework for Ambient Networks
- Policies used for access control
  - No policies for composition, which is the most important feature of AN
- Proof-of-concept prototype implemented
  - Important feedback for a new version of PBMAN



Paper Proposal

- To provide a PBM solution for Ambient Networks (AN), focusing on AN composition
  - Extension to the AN Architecture
- Expected contributions
  - An architecture for PBM for AN, built upon the previous architecture, based on P2P
    - Policies are first class citizens
  - New composition framework for AN
  - Modeling of a simple scenario
  - Policies for AN composition are proposed
  - Policies are written in XACML (extended)

PBMAN Architecture



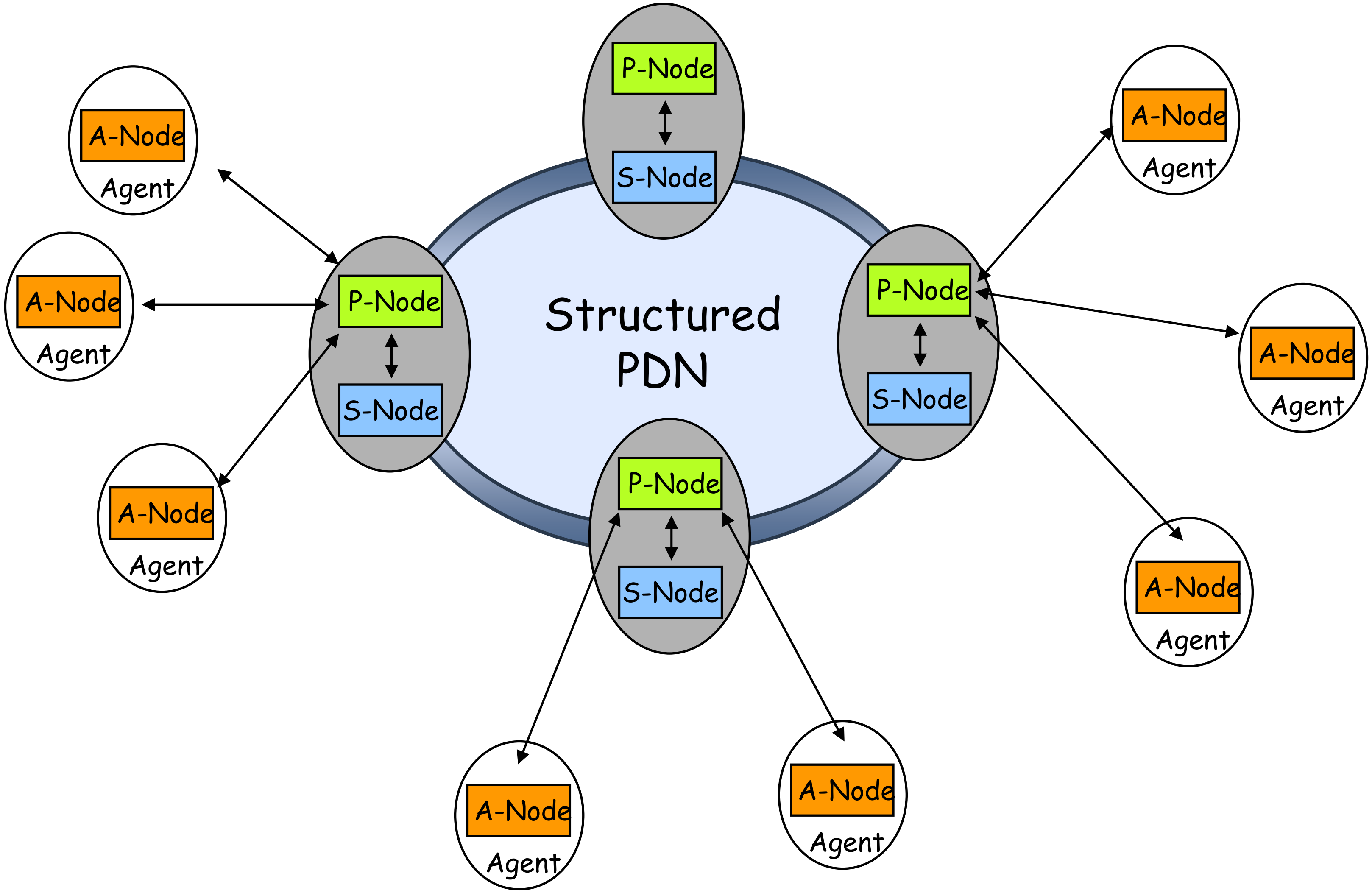
Networks and Nodes

- Policy Decision Network (PDN)
  - Policy Layer
  - Nodes (e.g. servers) interconnected by design
  - P-Nodes: management and policy decision tasks
- Storage Network (STN)
  - Storage Layer
  - S-Nodes: repository-specific nodes
- Agent Network (AGN)
  - Agent layer
  - A-Nodes: hosts, devices,… (PEPs)
- Nodes are not necessarily physical entities

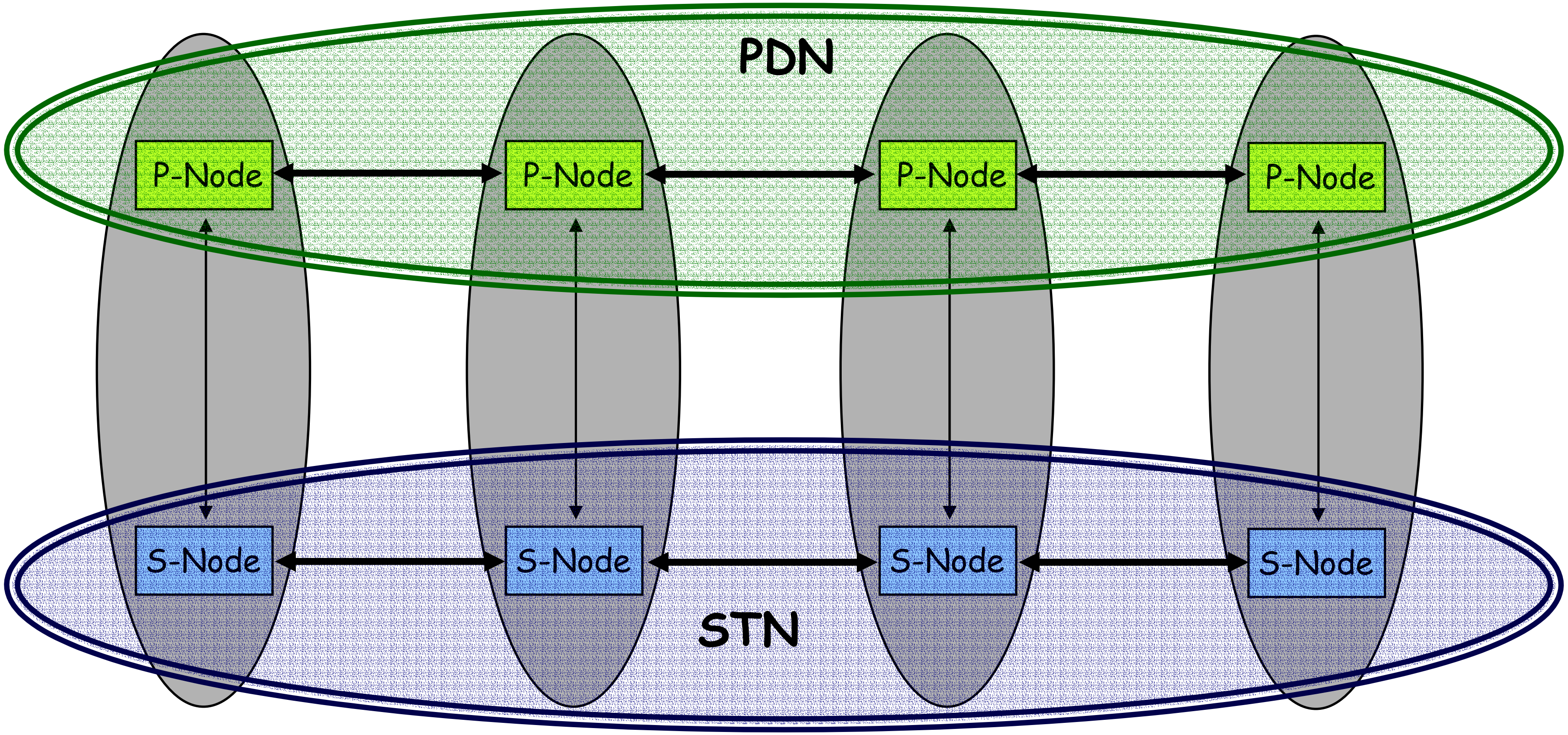
Composition Framework

- Different network entities have different composition requirements
- PBMAN identifies different composition classes to obtain efficient design and implementation
- Composition Dimensions
  - **Role**: Agent, Policy and Storage Compositions
  - **Scope**: Network, Node and Startup Compositions
- Examples:
  - Policy Network Composition, Agent Node Composition
- All of them controlled by policies

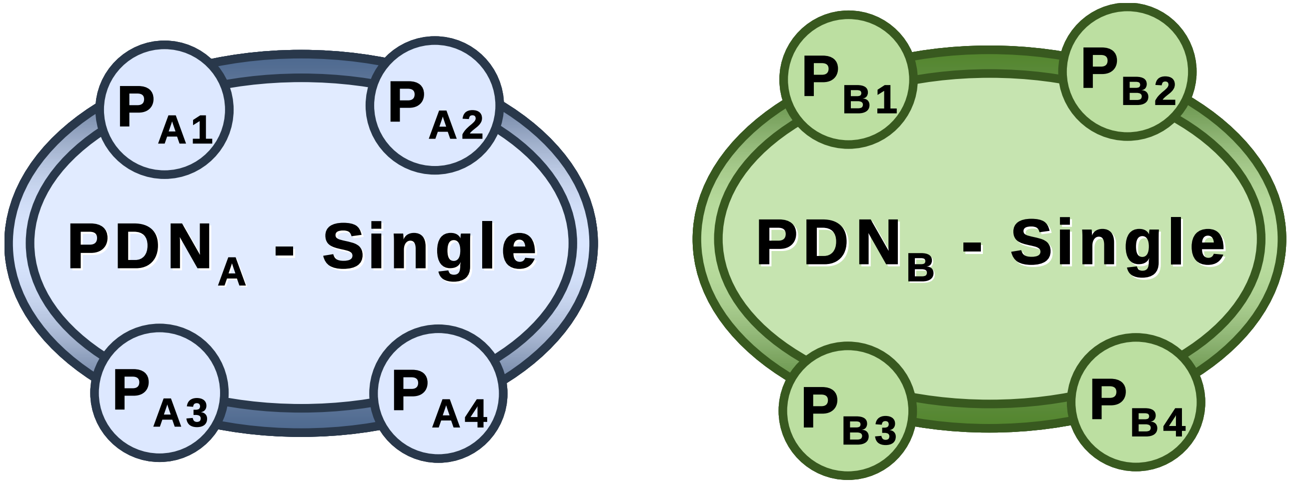
Structured PDN



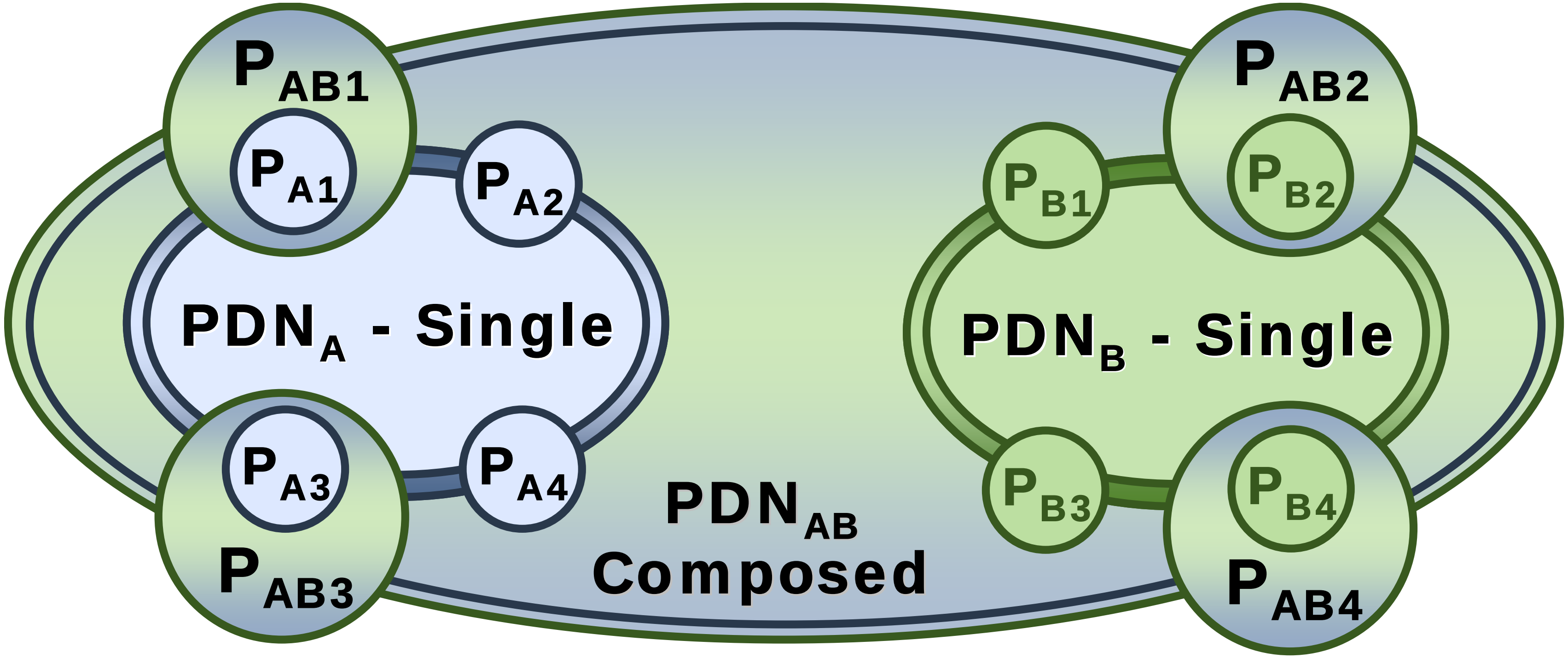
Policy and Storage Composition



Policy Network Composition - Before

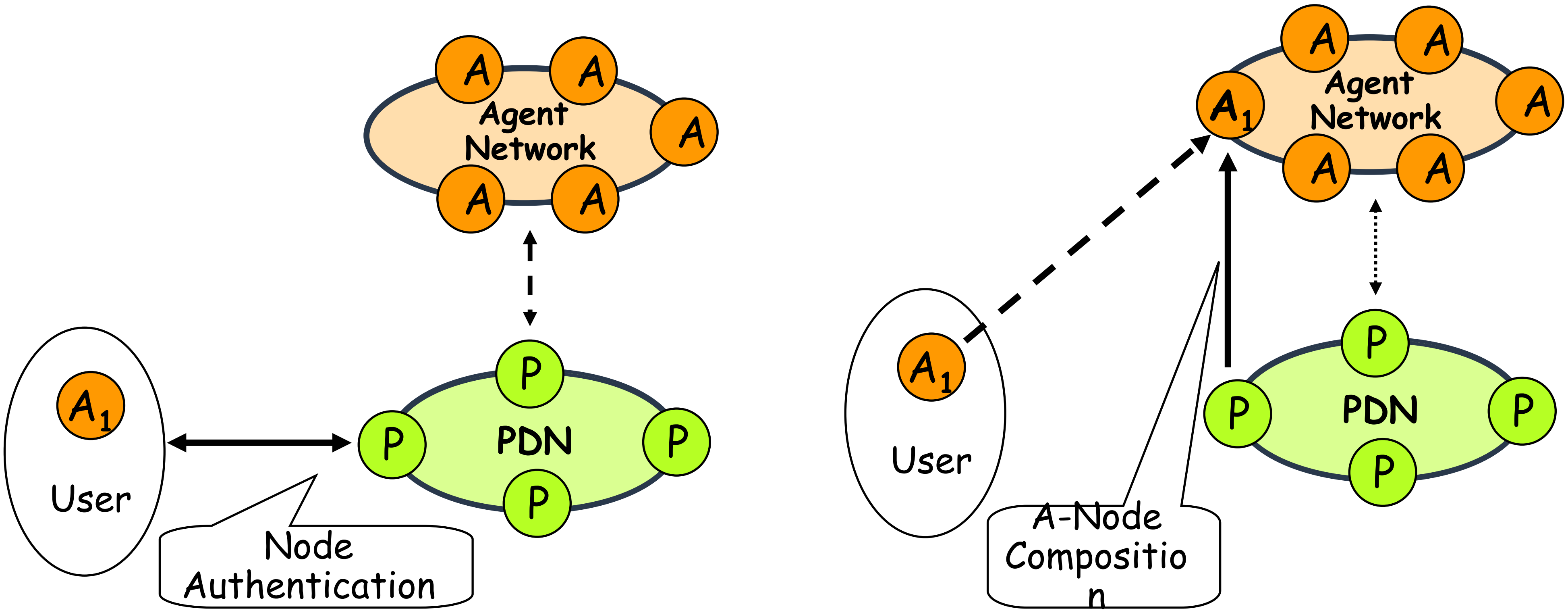


Policy Network Composition - After



- When networks get composed, policies of both networks are composed too

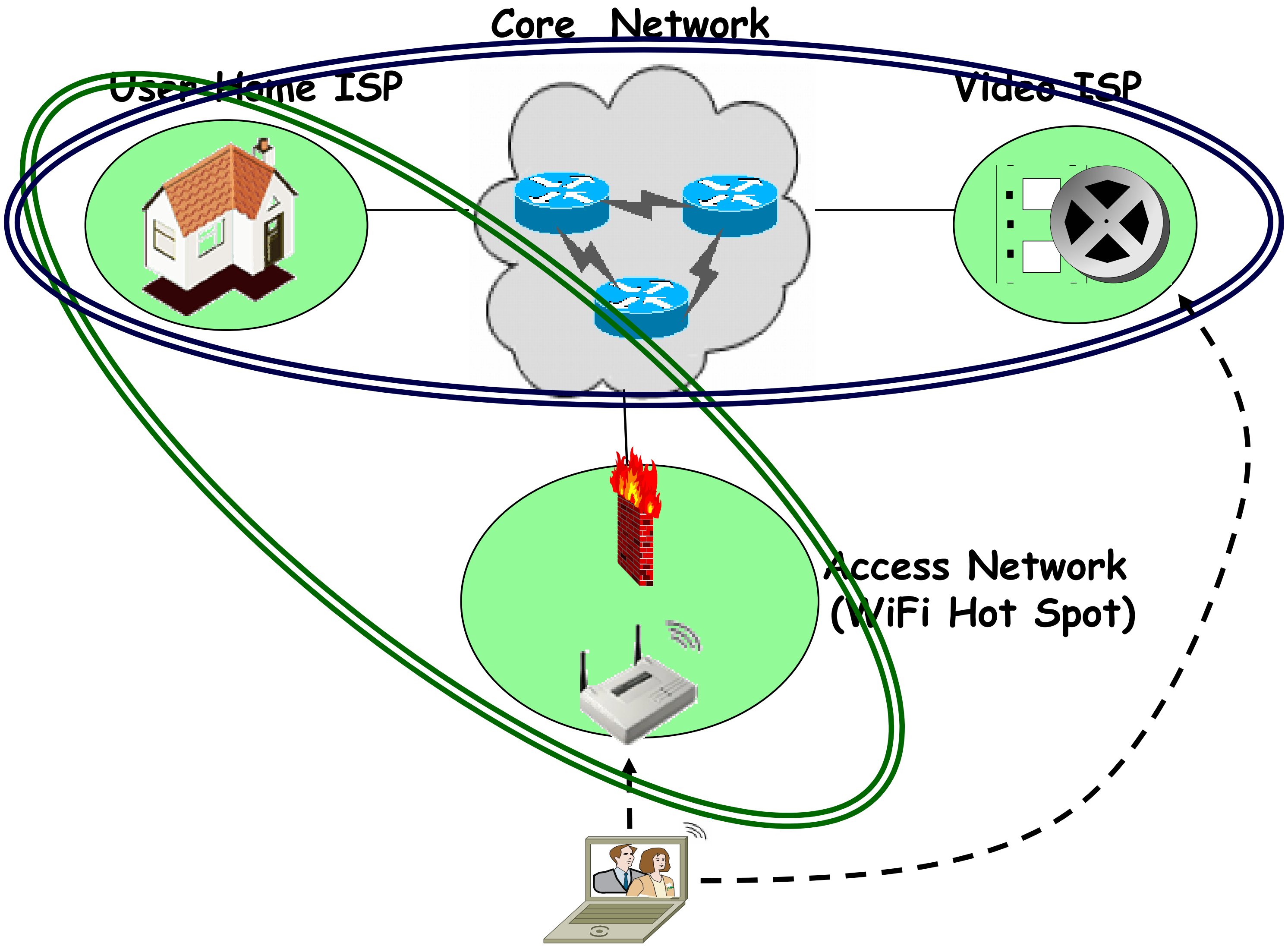
Agent Node Composition



Node Authentication

A-Node Compositio n

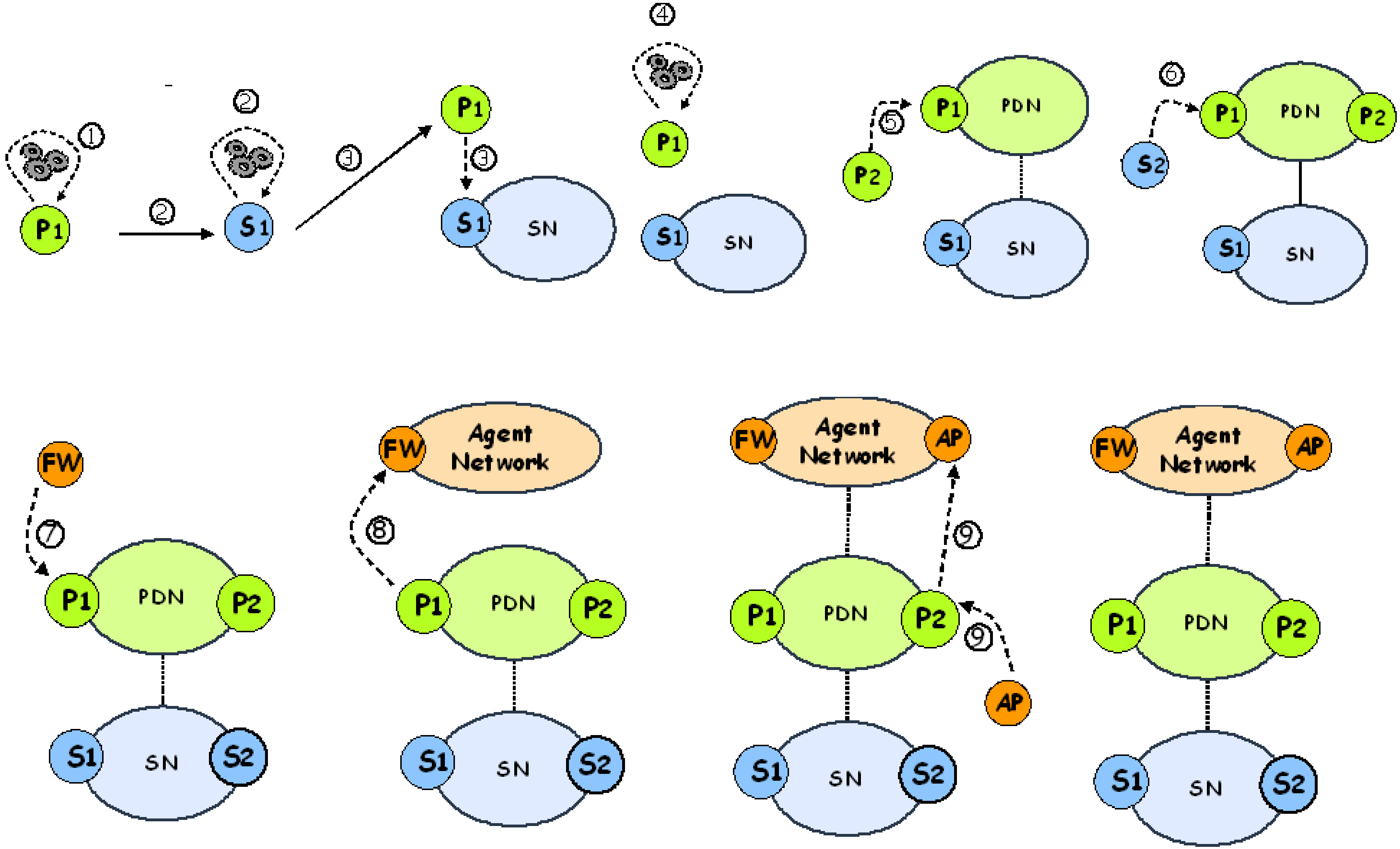
Scenario Modeling and Policies



Scenario: Characteristics

- Scenario comprised of two distinct phases
  - Bootstrapping all networks
  - Using services (network access and video)
- Compositions for bootstrap
  - Node and Startup compositions (policy, storage and agent)
- Composition for service usage
  - Network and Node compositions (policy and storage)
- Both involve the three layers of the architecture

Transaction for Bootstrapping (Wi-Fi access service)



Policies for Bootstrapping

(XACML policies – simplified syntax)

Policy P1; **Priority**: 1; **Type**: node-composition; **Effect**: Permit

Target: resource=access-agent-network

subject=any-node

action=compose

Condition: CA.agentNetUp(access-agent-network)

Processing: CA.addAttribute

(access-agent-network.ca-dynamic-nodes,

$request.node)

Obligation: n/a

Policies for Bootstrapping

Policy P2; **Priority**: 1; **Type**: node-composition; **Effect**: Permit

Target: resource=access-agent-network
subject=any-node
action=compose

Condition: !CA.agentNetUp(access-agent-network)

Processing: Composition.**request**
(resource= access-agent-network;
subject=$request-node; action=**compose**;
role=**agent**; scope=**startup**)

Obligation: n/a

Policies for Service Usage

Policy P4; **Priority**: 0; **Type**: access-control; **Effect**: Permit

Target: resource=any-service;
subject=any-subject;
action=start

Condition: $request.an <> $CA.id &&
!CA.policyNetUp($request.an,$CA.id)

Processing: **Composition.request** (resource=$request.an;
subject=$CA.id; action=**compose**; role=**policy**;
scope=**network**)

Processing: **Service.request** (resource=$request.service;
subject=$request.subject;
action=$request.action)

Obligation: n/a

Policies for Service Usage

Policy P6; Priority: 2; Type: node-composition; Effect: Permit

Target: resource=video-agent-network;
subject=any-node;
action=compose

Condition: CA.agentNetUp(video-agent-network) &&
CA.isUser($request.node) &&
video-agent-network.current-users <
video-agent-network. max-user

Processing: CA.addAttribute(video-agent-network.
ca-dynamic-users, $request-node)

Processing: CA.addAttribute(video-agent-network.
ca-current-users, 1)

Obligation: n/a

Current Status and Future Work

- Current Status
  - Most specifications are done
  - Prototype development is being finished (p2p storage)
  - Evaluation will begin soon
  - Transactions and policies have been rewritten
- Future Work
  - Support for conflict resolution
  - User-friendly PMT (under development)
  - Add support for mobility and wireless users

Conclusions

- PBMAN2: PBM framework for Ambient Networks
    - Current concepts evolve from an early version
- PBMAN now uses XACML
- Simple scenario modeled and policies written
- Lessons learned (so far)
    - Putting policies to work needs more effort than just writing policies
        - Framework needed with the right “slots” for policies
    - The problem is in the details
        - Implementation needed to be down-to-earth
    - Writing policies is not easy
        - A good Policy Management Tool is needed

XACML-Based Composition Policies for Ambient Networks

Thank You!

This work was supported by the Research and Development Centre, Ericsson Telecomunicações S.A., Brazil