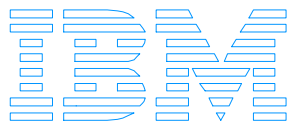




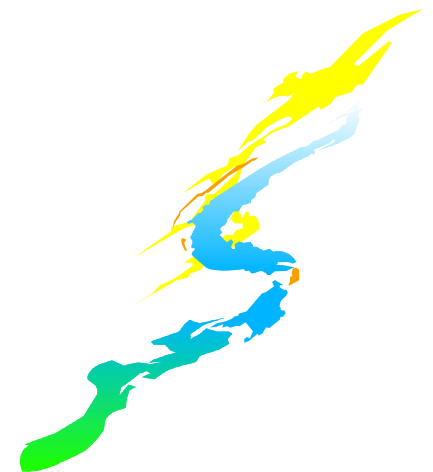
Network Processors, Enablers for Programmable Networks

OpenSIG 2001, September 24/25

Patrick Droz



Zurich Research Laboratory



Outline

- Introduction to network processors
 - Example IBM PowerNP 4GS3
- Router architecture changes
 - Impact on the hardware
 - Impact on the software
- Standards efforts
- Network processors & active networking
- Summary



Network Processors

- Dedicated chips that can do packet processing in routers and switches mainly associated with data forwarding.
 - Longest prefix matching in h/w (forwarding)
 - Scheduling / policing (QoS)
 - CRC calculation (header processing)
 - Encryption (security)
 - Classification / filtering (firewalling)
 - Programmability

Network Processors (cont.)

- Product offerings must include:
 - Hardware reference platform
 - Software reference platform
 - Rich set of tools
 - Compilers
 - Debuggers
 - Simulators
 - Profilers

PowerNP 4GS3

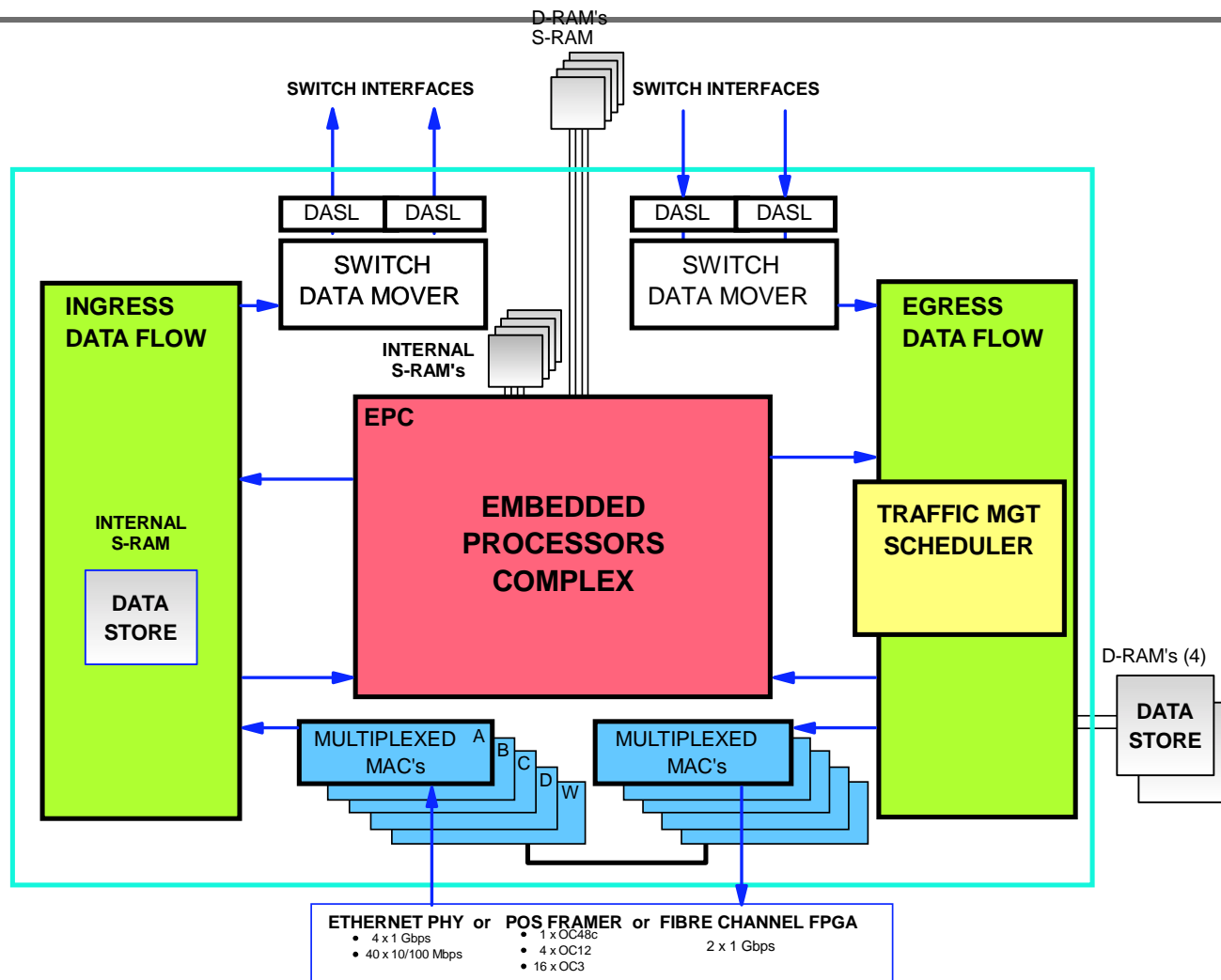
- Component Integration
 - Embedded Processor Complex
 - Embedded PowerPC
 - Dataflow
 - Scheduler
- Packet Processing
 - Processing Power
 - 16 Pico-Processors, 32 Threads
 - Deep Packet Processing
 - 8 Programmable Search Engines
 - Multi-layer Scalability

PowerNP 4GS3 (cont.)

- Programmability
 - High and Low Level APIs
 - Write software without worrying about chip specifics
 - Support of 3rd party software

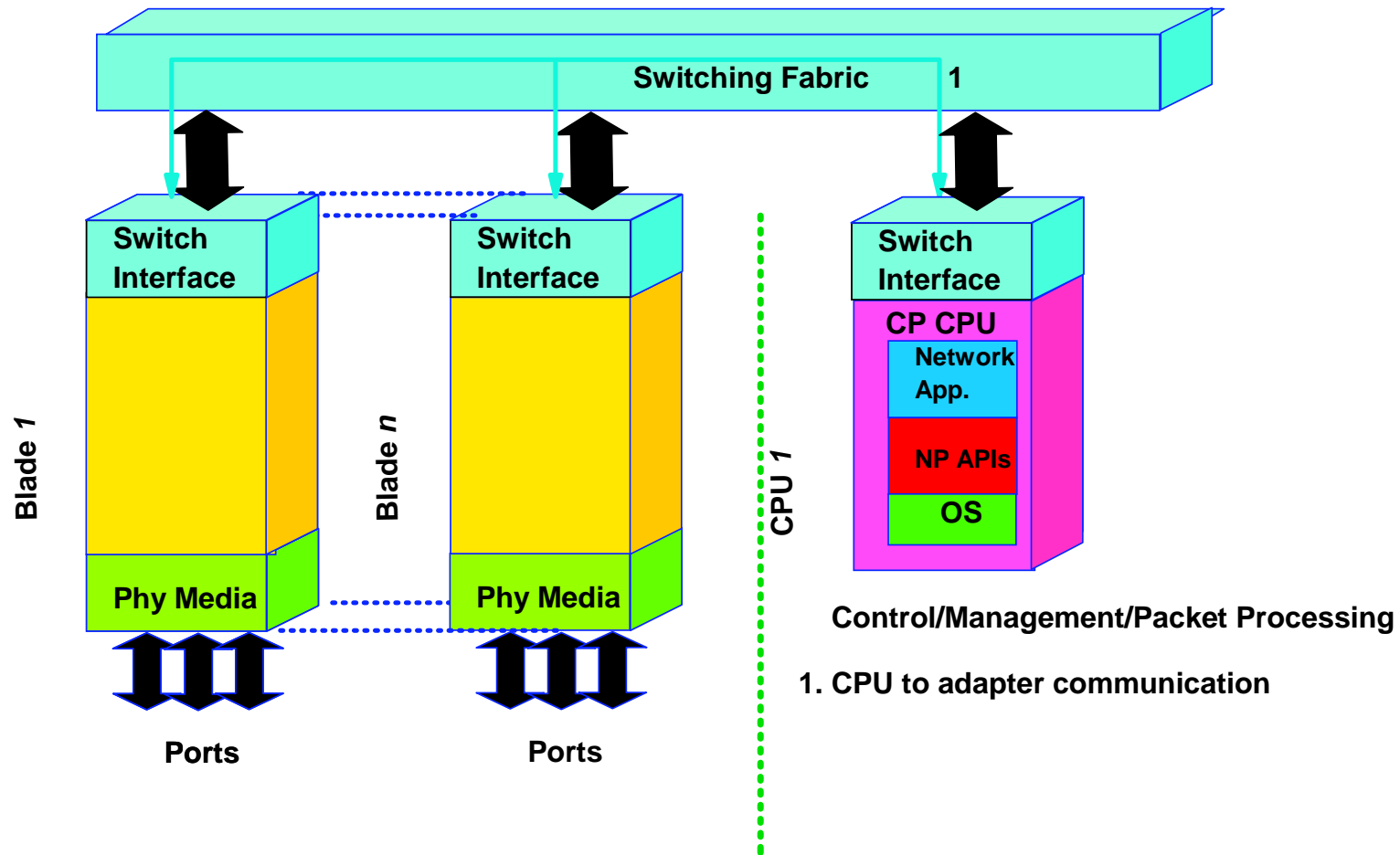


PowerNP 4GS3

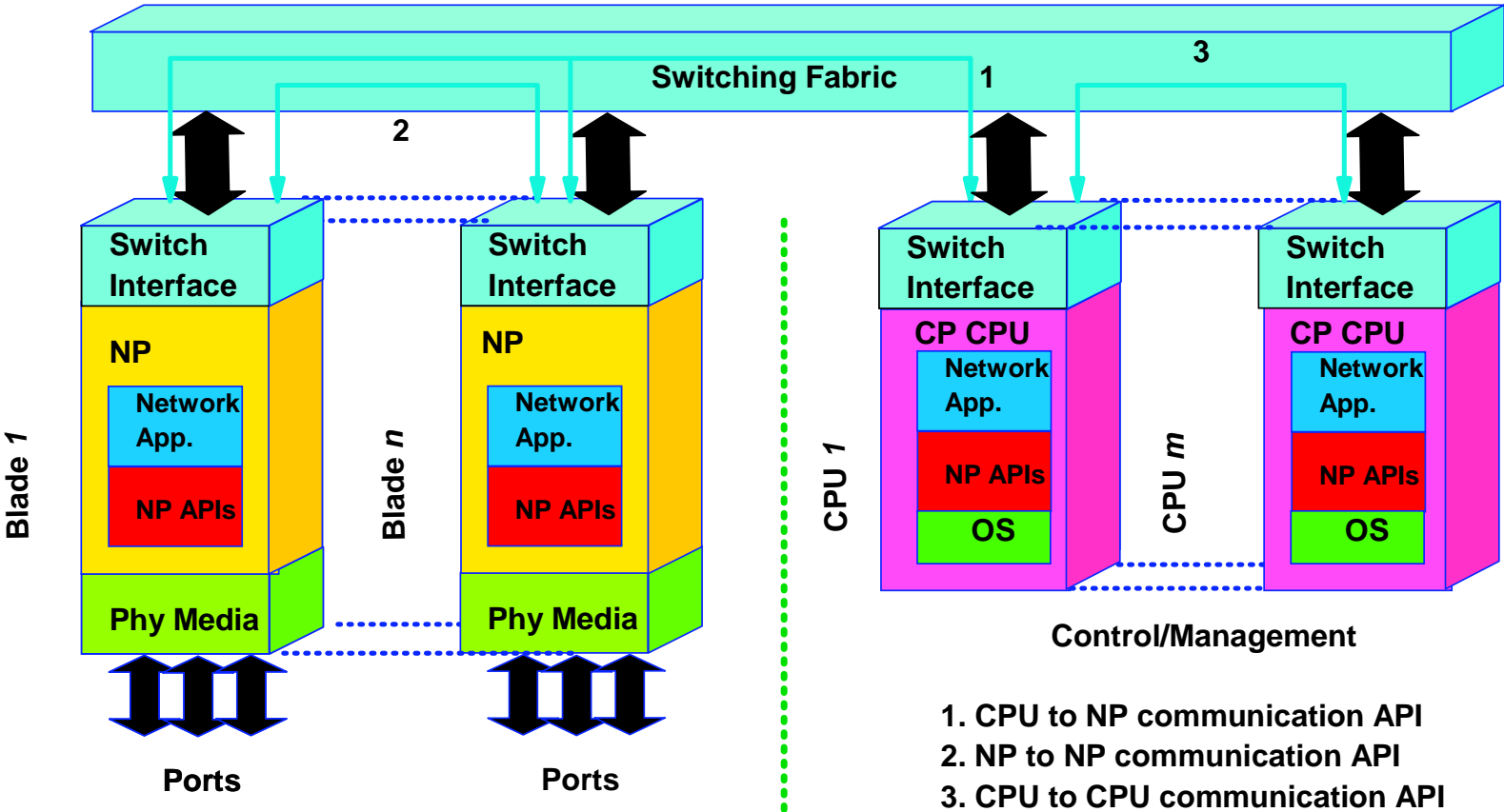


Zurich Research Laboratory

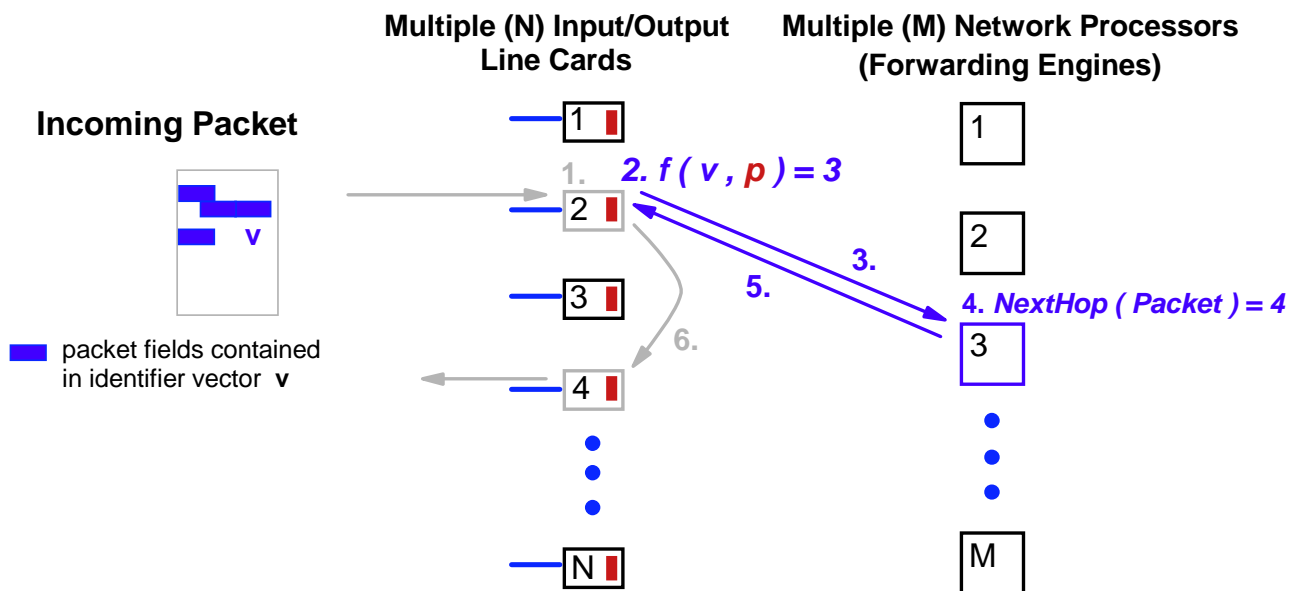
Centralized Architecture



Distributed Architecture



Packet Processing Scheme Abstraction



Packet processing steps:

1. packet parsed, buffered
2. location of appropriate processor determined by computing the mapping f , a function of the identifier vector v and the load balancing vector p ($f(v, p) = i$); $p = (p_1, p_2, \dots, p_m)$ is a vector defining size of fraction of identifier vector space mapped to each processor
3. request sent to appropriate processor for packet processing
4. packet information processed ($NextHop(Packet) = j$)
5. resolution info sent back to the originating node
6. upon desired treatment applied, packet forwarded towards the indicated output



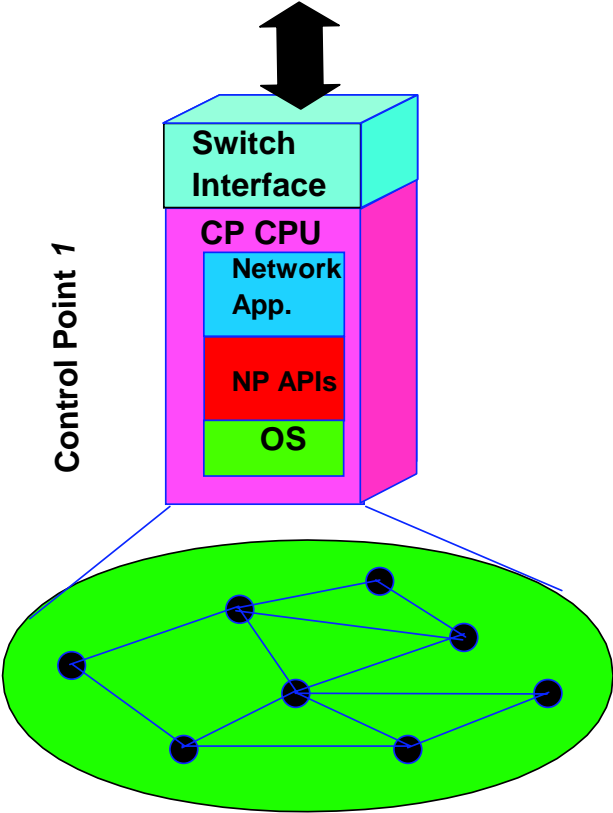
Impact on the Software

- Client / server environment
- Increased heterogeneity
- Increased control and management communication
- Distributed algorithms
- Partitioning
- Higher complexity
- Adaptive software stacks

Distributed Architecture

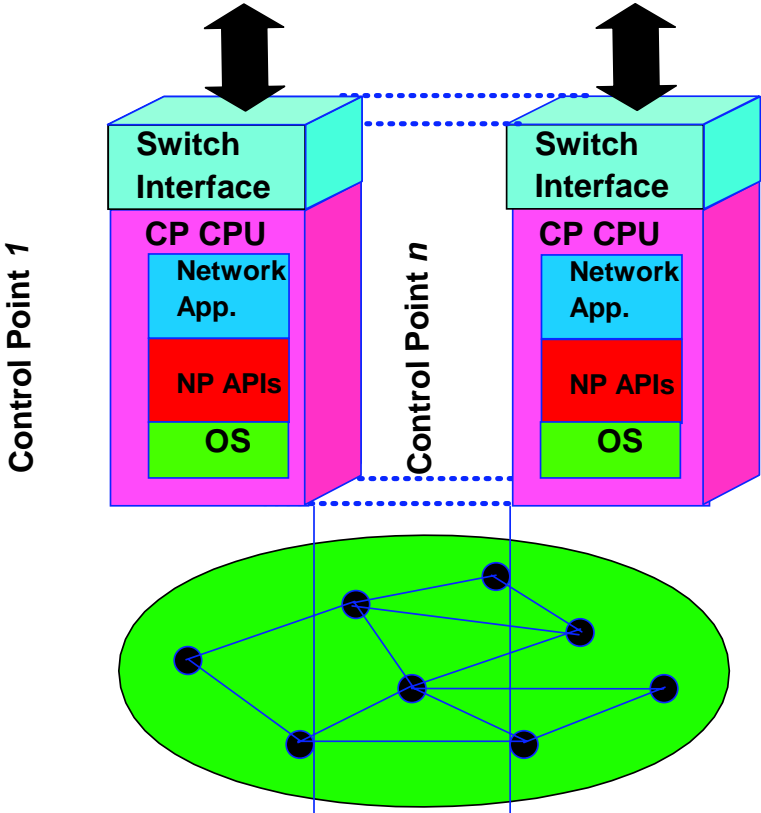
- APIs:
 - CPU to NP,
 - NP to NP,
 - CPU to CPU.
- High-level API:
 - Routing table management and lookup.
- Low-level API:
 - Packet classification algorithm.

Distributed Control Point Example



CPU 1

central topology DB & centralized algorithms



CPU 1

CPU n

distributed topology DB & distributed algorithms



Network Processing Forum (NPF)

■ Mission:

- Leverage the usage of network processors technologies by means of:
 - standards
 - testing
 - benchmarking
 - education

■ NPF = (CSIX + CPIX) in February 2001



Zurich Research Laboratory

ForCES at the IETF

- A protocol between a control element CE and a forwarding element FE
 - h/w capability discovery
 - control and management
 - automatic service deployment
 - based on h/w abstraction
- Currently working on the requirements
 - Design team in place
- First WG meeting at the last IETF in London

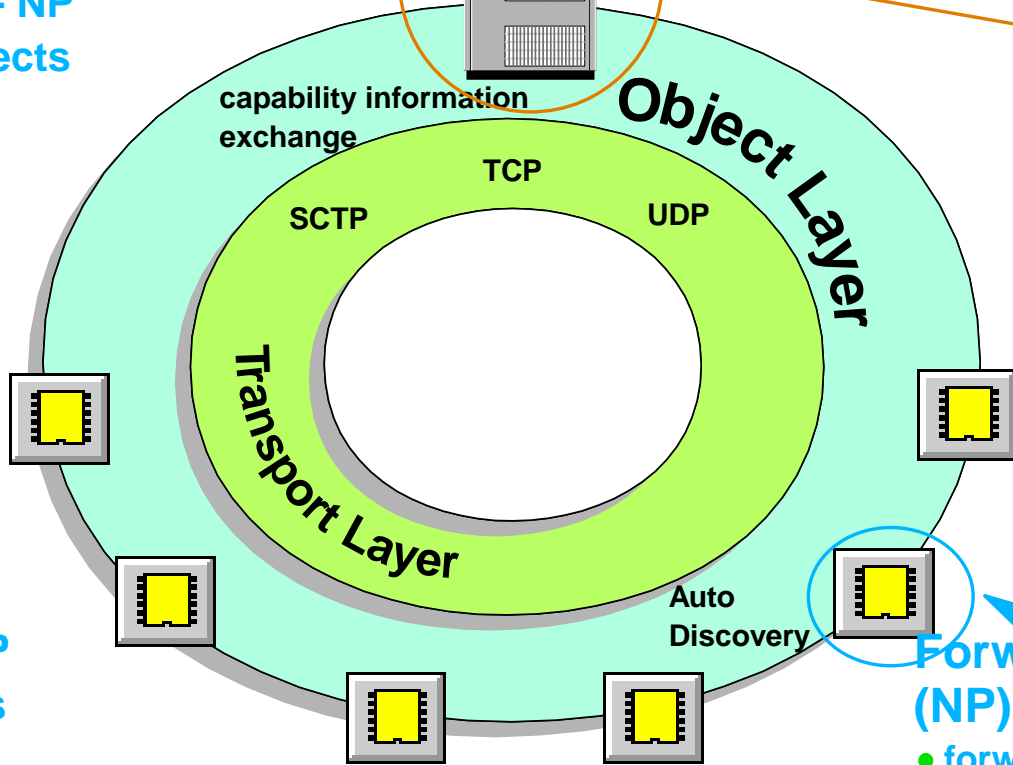
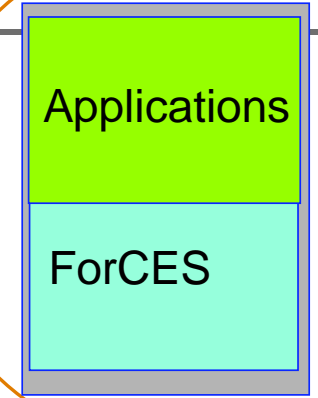


ForCES - Forwarding and Control Element Separation

Control Element (CP)

- routing protocols
- admission control
- signaling

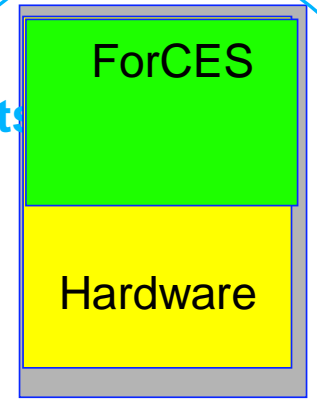
CP - NP objects



Forwarding Elements (NP)

- forwarding
- queuing
- frame alteration
- intserv, diffserv QoS

NP - NP objects



Zurich Research Laboratory

NPs & Active networking

- Open interfaces
- Standards
- Programmability
- h/w assists
- Performance
- Distributedness
- PnP requirements



Automatic Service Deployment

■ Network infrastructure

- ▶ heterogeneous & programmable
- ▶ composed of NPs, FPGAs, ASICs, software, etc.
- ▶ combinations of the above

■ Network services

- ▶ key differentiation for ISPs
- ▶ still slow to deploy: manual install and configure

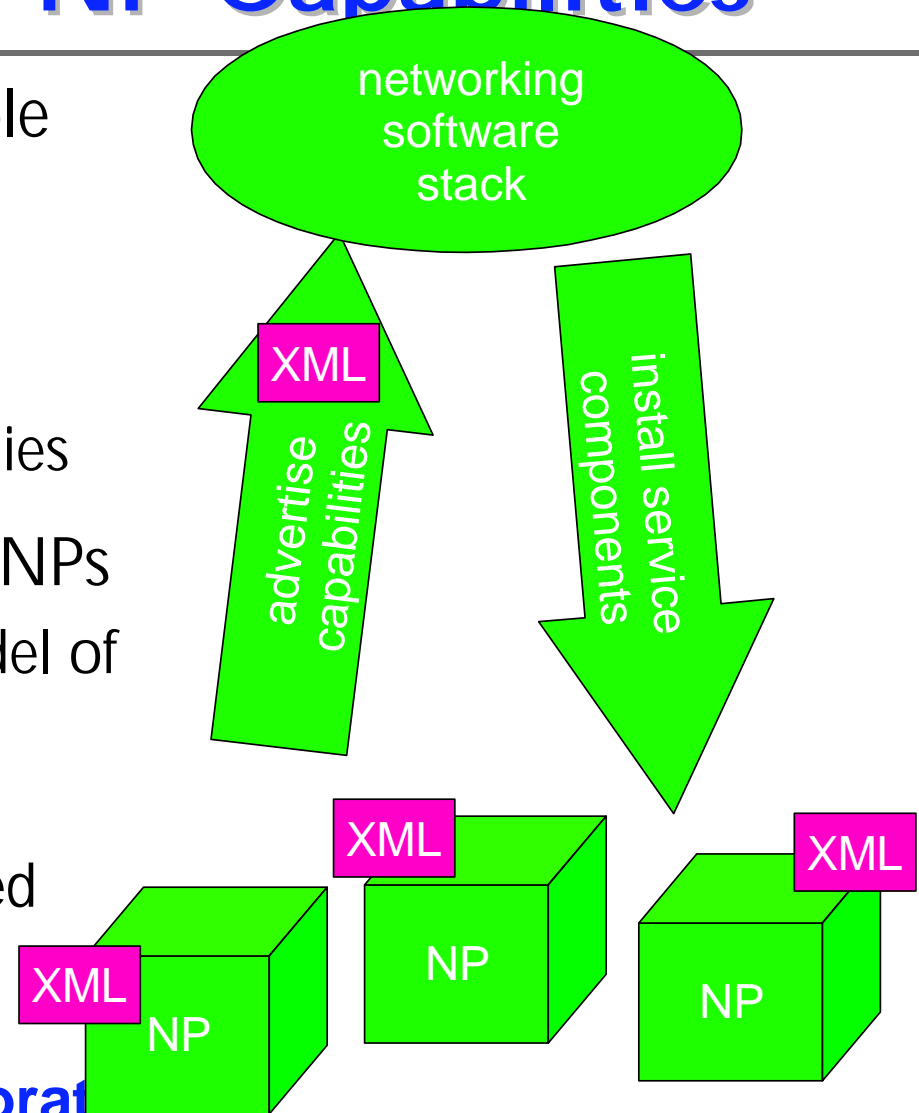
➔ provide automated service deployment

- ▶ make "best use" of capabilities of the infrastructure
- ▶ automatically organize installation and configuration



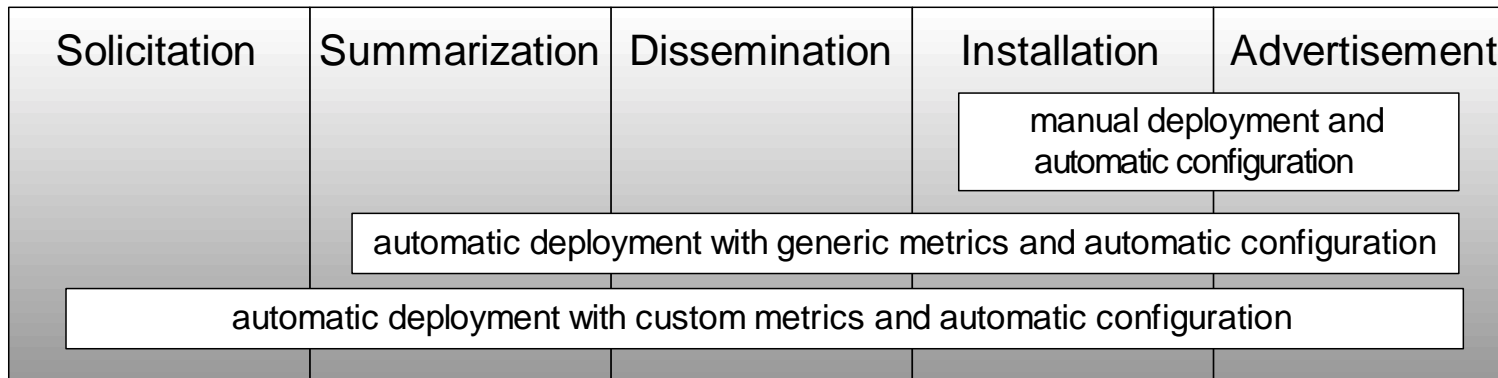
Auto-Discovery of NP Capabilities

- Plug & Play (PnP) for programmable networking components:
 - h/w has chameleon-like behavior
 - PnP NP picocode, etc.
 - make best use of available capabilities
- s/w stack services poll underlying NPs
 - NP provides an XML abstract model of architecture showing:
 - performance
 - Control & Traffic APIs supported
 - hardware capabilities



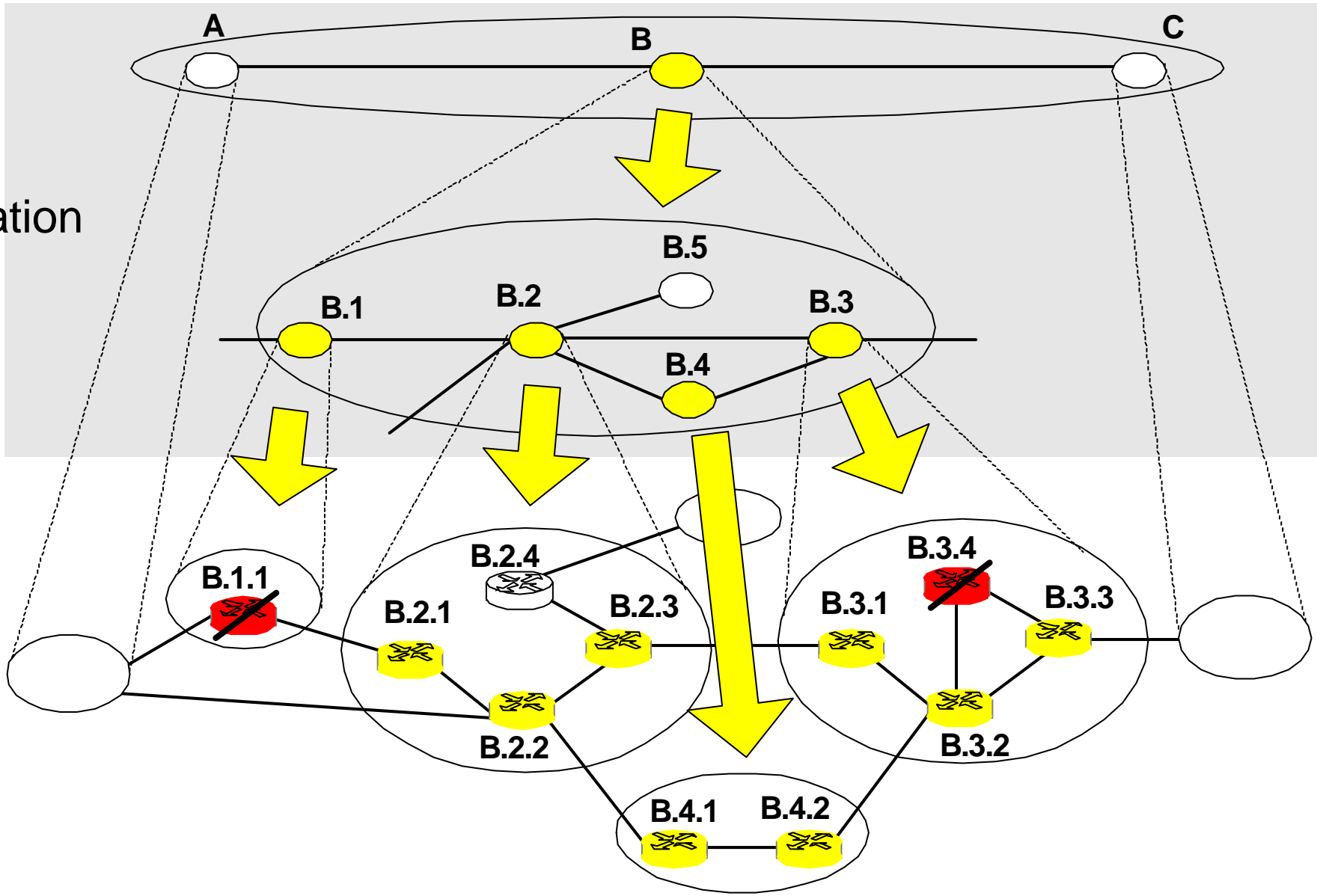
Service Deployment Framework

- control-theory based deployment
- extension of QoS routing to service routing
- examples of three categories of services:
 - path-based (sparse or continuous): diffserv
 - node-based: webcache
 - path- and node-based: VPN (QoS + encryption)



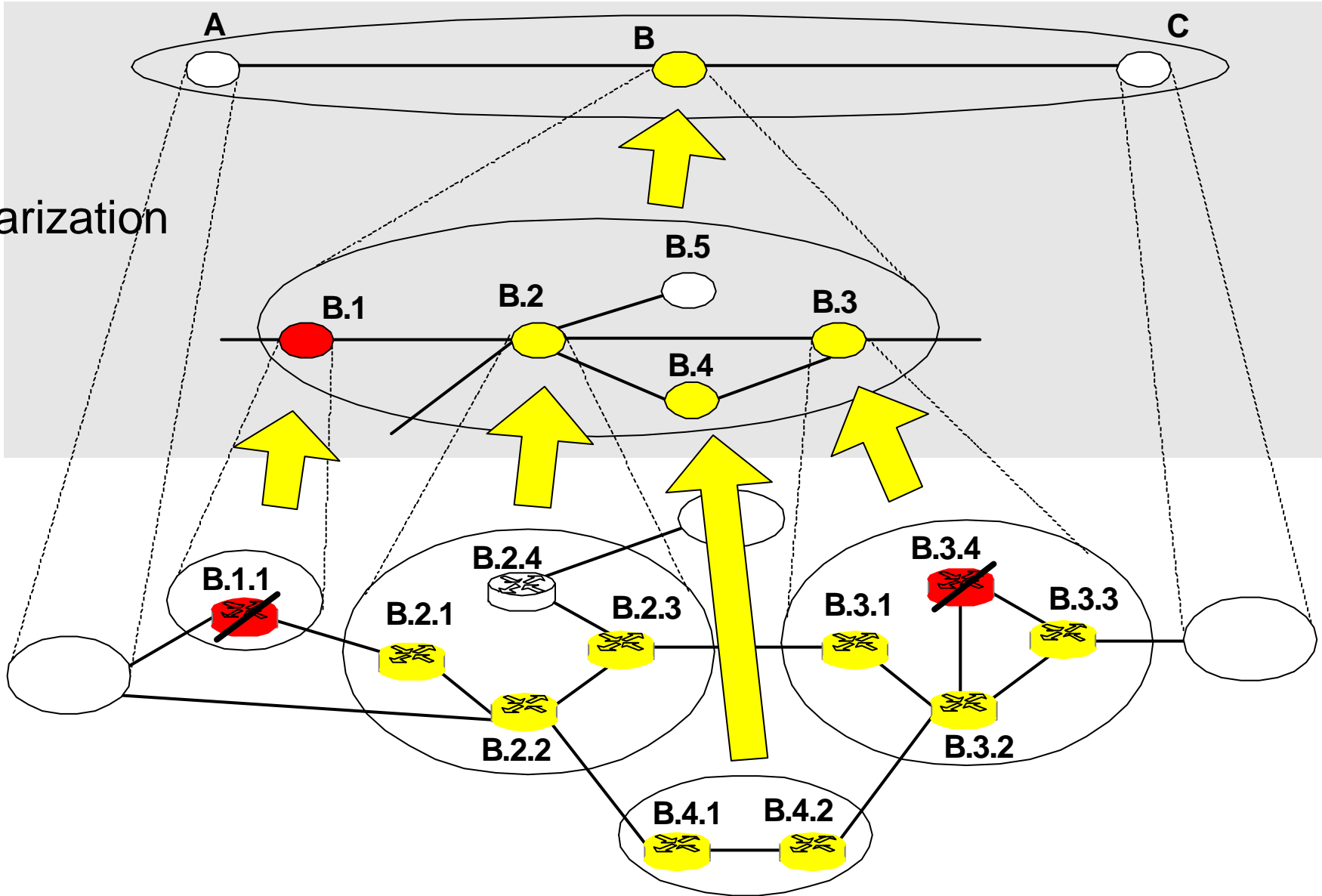
Example of path-based deployment

solicitation



Example of path-based deployment

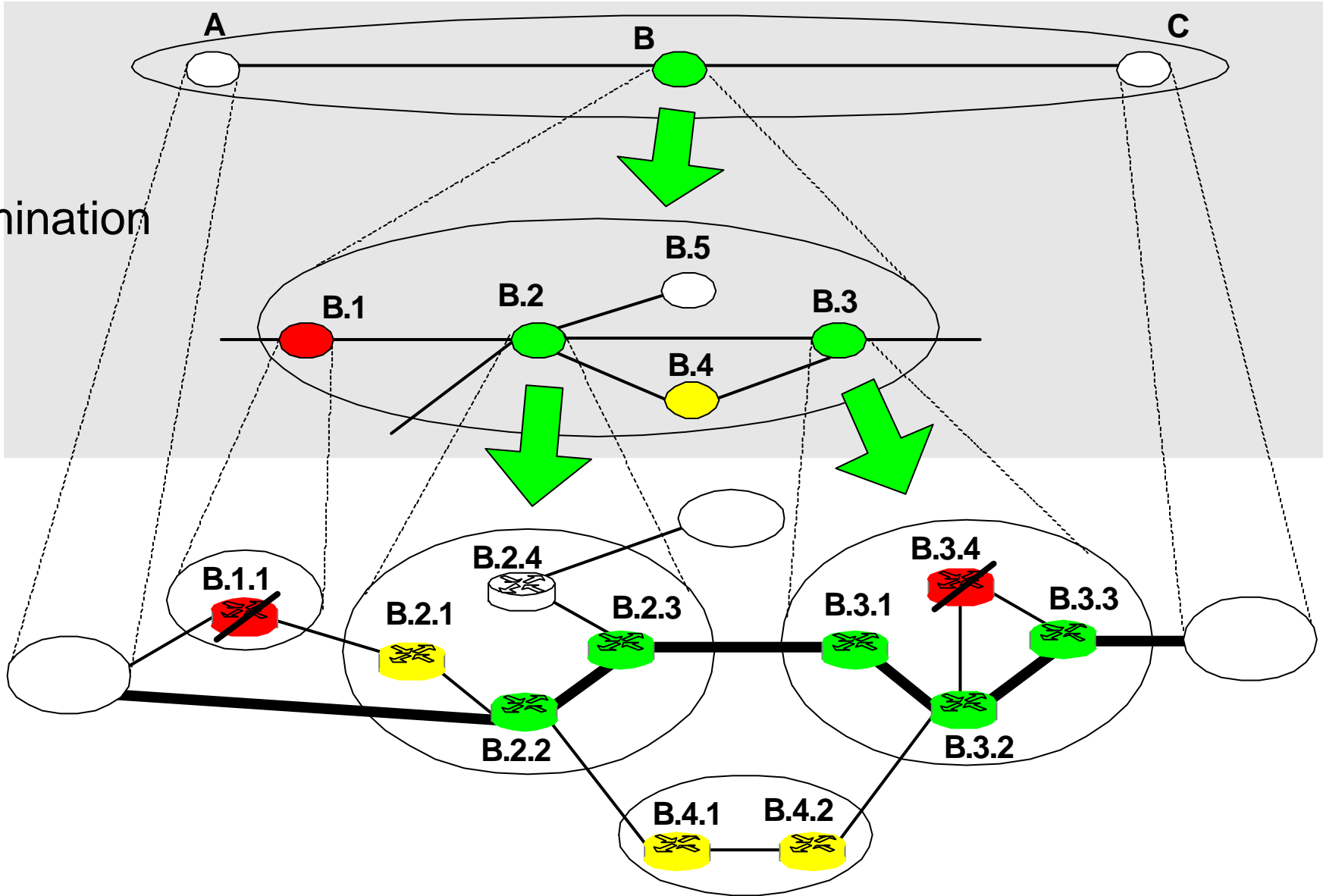
summarization



	solicited, capable, and enabled router		solicited and capable but not enabled		solicited but not capable		not solicited
---	--	---	---------------------------------------	---	---------------------------	---	---------------

Example of path-based deployment

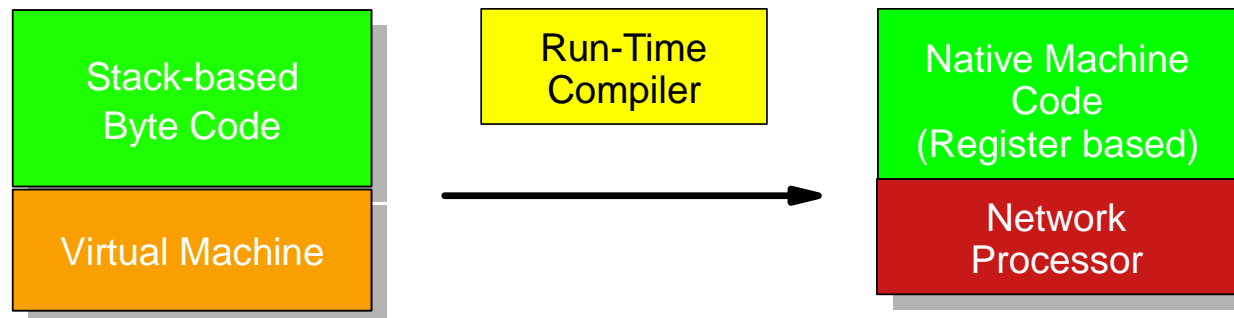
dissemination



	solicited, capable, and enabled router		solicited and capable but not enabled		solicited but not capable		not solicited
---	--	---	---------------------------------------	---	---------------------------	---	---------------

An Example of Active Networking on Network Processors

- Analysis of run-time compilers for active networks on network processors.



- Environment:
 - Based on an existing byte-code language (SNAP) that uses the capsule approach.
 - A modified resource bound allows controlled looping.
- Current status: Active Networking Sandbox (virtual machine) implementation and run-time compiler on a PowerNP.

An Example of Active Networks (cont.)

■ Results:

- Reduction of the interpretation overhead from the virtual machine.
- Program loops are in favour of the run-time compilation.
- Due to the underlying specialized hardware, execution time of byte code instruction varies.

■ Further improvements

- In homogeneous environments:
The compilation overhead can be further reduced if compiled code is cached either at nodes (i.e. not removed from instruction memory) or cached inside the packet.
- Static analysis of active network programs could help to determine optimal initial resource bounds.

Summary

- Introduction into NPs
- Router architecture changes
- More complex software architecture
- Standards activities
- NPs & AN
- Examples

