Dynamic MPEG-4 service setup in Multi-protocol Access Inter Domain architecture

Giodi Giorgi

Università degli Studi di Pisa

Dipartimento di Ingegneria della Informazione: Elettronica, Informatica, Telecomunicazioni

Gruppo di Ricerca in Reti di Telecomunicazioni

wwwtlc.iet.unipi.it

Outline

- The Multi-protocol Access Inter Domain architecture
  - The Common Open Policy Service and its extensions
  - The seamless inter-domain QoS-IP services
- Inferring QoS-IP parameters from multimedia applications in the MAID architecture
  - The QoS-IP generalized syntax
- Inter-working between the MPEG-4 DMIF and the MAID QoS-IP signaling
  - MPEG-4 architectural overview
  - MPEG-4 DMIF: message flow
  - DMIF IWU implementation

Outline

- The Multi-protocol Access Inter Domain architecture
  - The Common Open Policy Service and its extensions
  - The seamless inter-domain QoS-IP services

- Inferring QoS-IP parameters from multimedia applications in the MAID architecture
  - The QoS-IP generalized syntax

- Inter-working between the MPEG-4 DMIF and the MAID QoS-IP signaling
  - MPEG-4 architectural overview
  - MPEG-4 DMIF: message flow
  - DMIF IWU implementation

QoS requirements

From the user perspective:

- the **dynamism** (e.g. the service should last as long as the user needs),
- the **tailoring** (e.g. the network resources allocated for the service should fulfill exactly the end-user requirements),
- a seamless **integration** (e.g. the mechanisms involved in QoS support should be transparent to end-user applications).

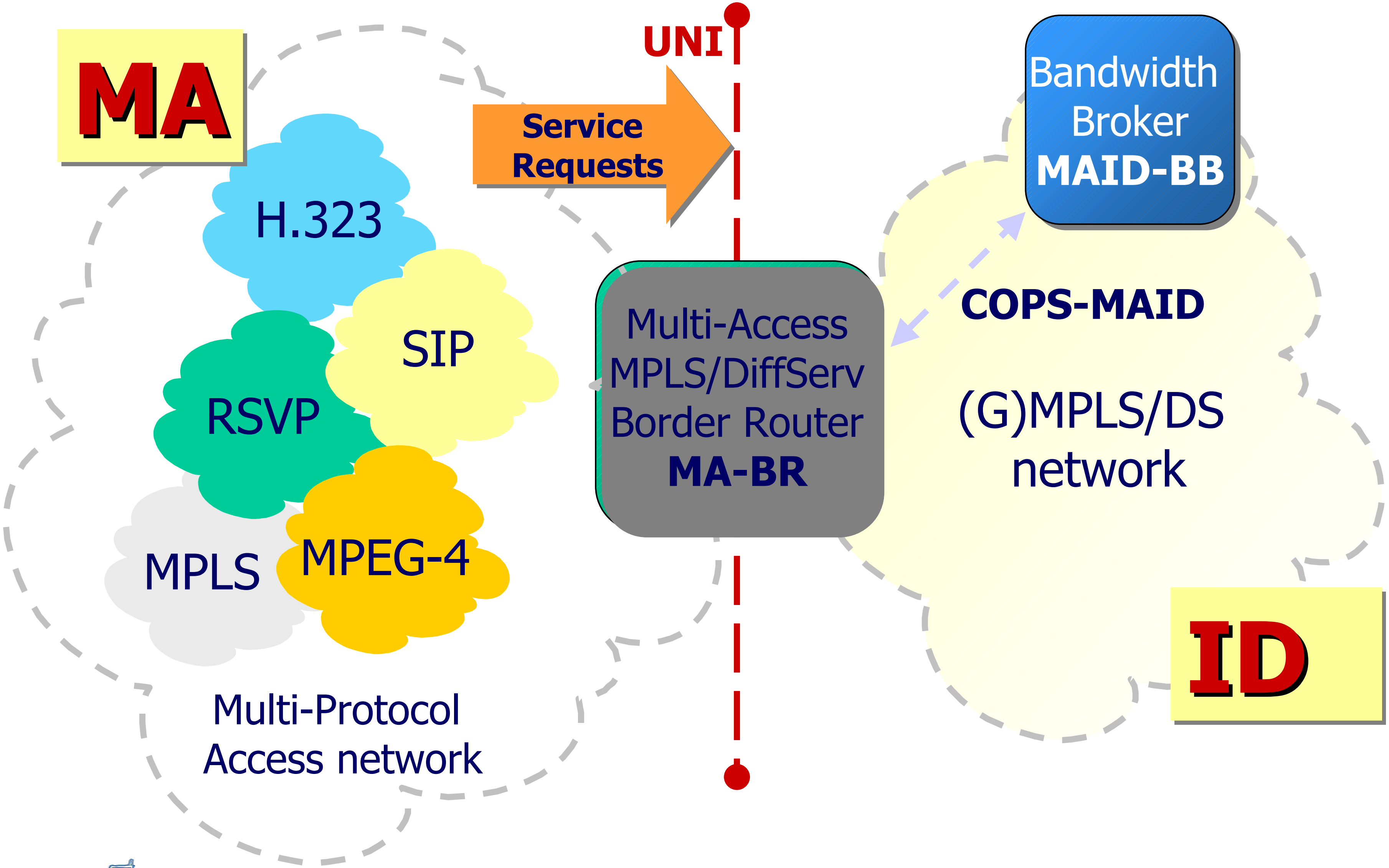
… indeed, the main obstacles

- the **different backbone** networks technologies (e.g. DiffServ, MPLS, IPoATM, etc.), which make hard to guarantee end-to-end QoS, above all when the service has to be deployed across different administrative domains
- the **number of different protocols** used in the access networks (e.g. RSVP, H.323, SIP, MPEG-4..), which implies a per-service/per-protocol User-Network-Interface (UNI)
- … and the required QoS is often not explicitly declared in the access signaling flow (e.g. H.323)

From the Service Provider perspective:

- **interoperation of adjacent domains** with the same or different technologies, which implies a Network-to-Network-Interface (NNI),
- **interoperation of equipments** from different vendors.

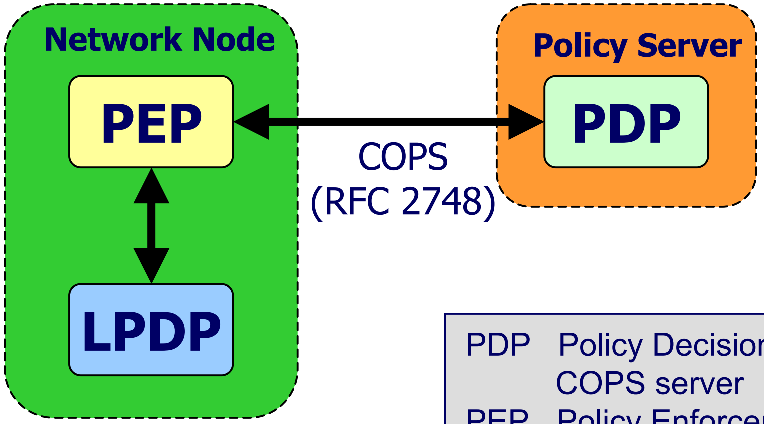
Multi-Access Inter-Domain



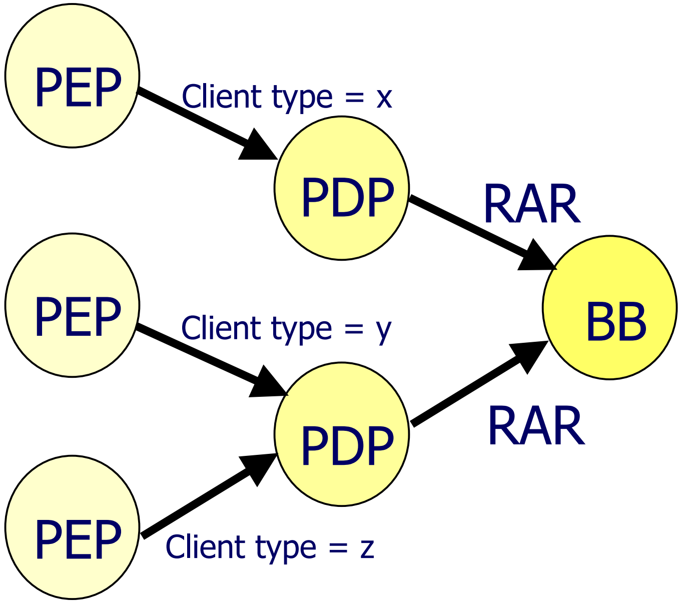
MA
UNI
Service Requests
H.323
SIP
RSVP
MPLS
MPEG-4
Multi-Protocol Access network
Multi-Access MPLS/DiffServ Border Router **MA-BR**
Bandwidth Broker **MAID-BB**
COPS-MAID
(G)MPLS/DS network
ID

The COPS protocol

- Different applications using different protocols → **different client** types.
- The trend to define a new client type for each access network protocol results in a hard limit to the system **scalability** (duplication of the states installed both in the COPS client and in the COPS server).
- A possible **solution**: the cluster of COPS server, each supporting one or few client-types.
- Disadvantage: all these COPS servers have either to exchange management information to perform a coherent resource allocation or should refer to a higher level ”omniscient” BB.

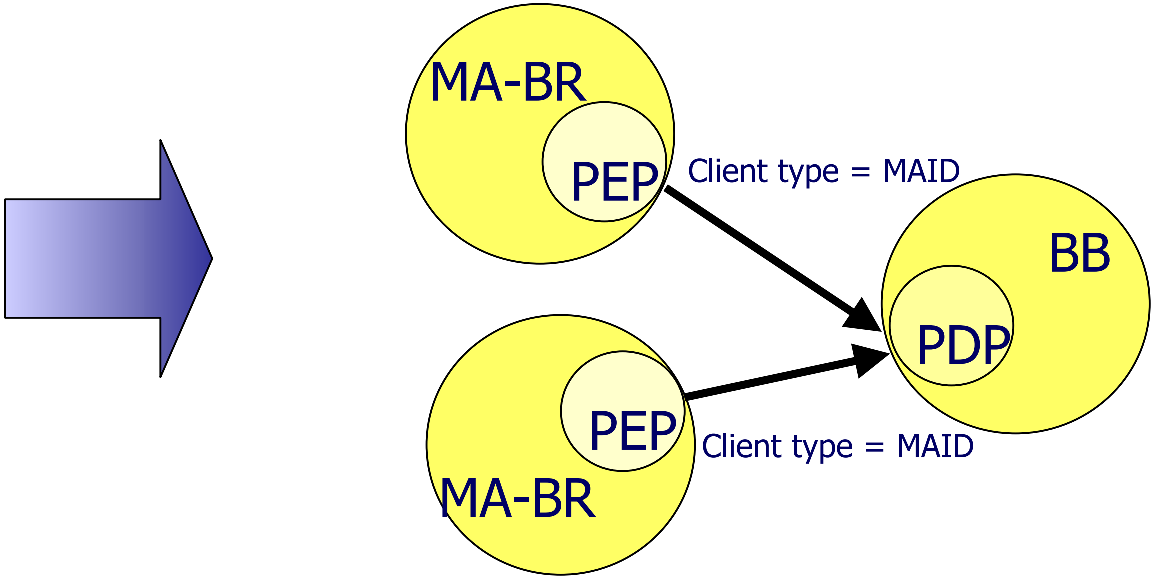


PDP Policy Decision Point,
COPS server
PEP Policy Enforcement Point,
COPS client



COPS-MAID extensions

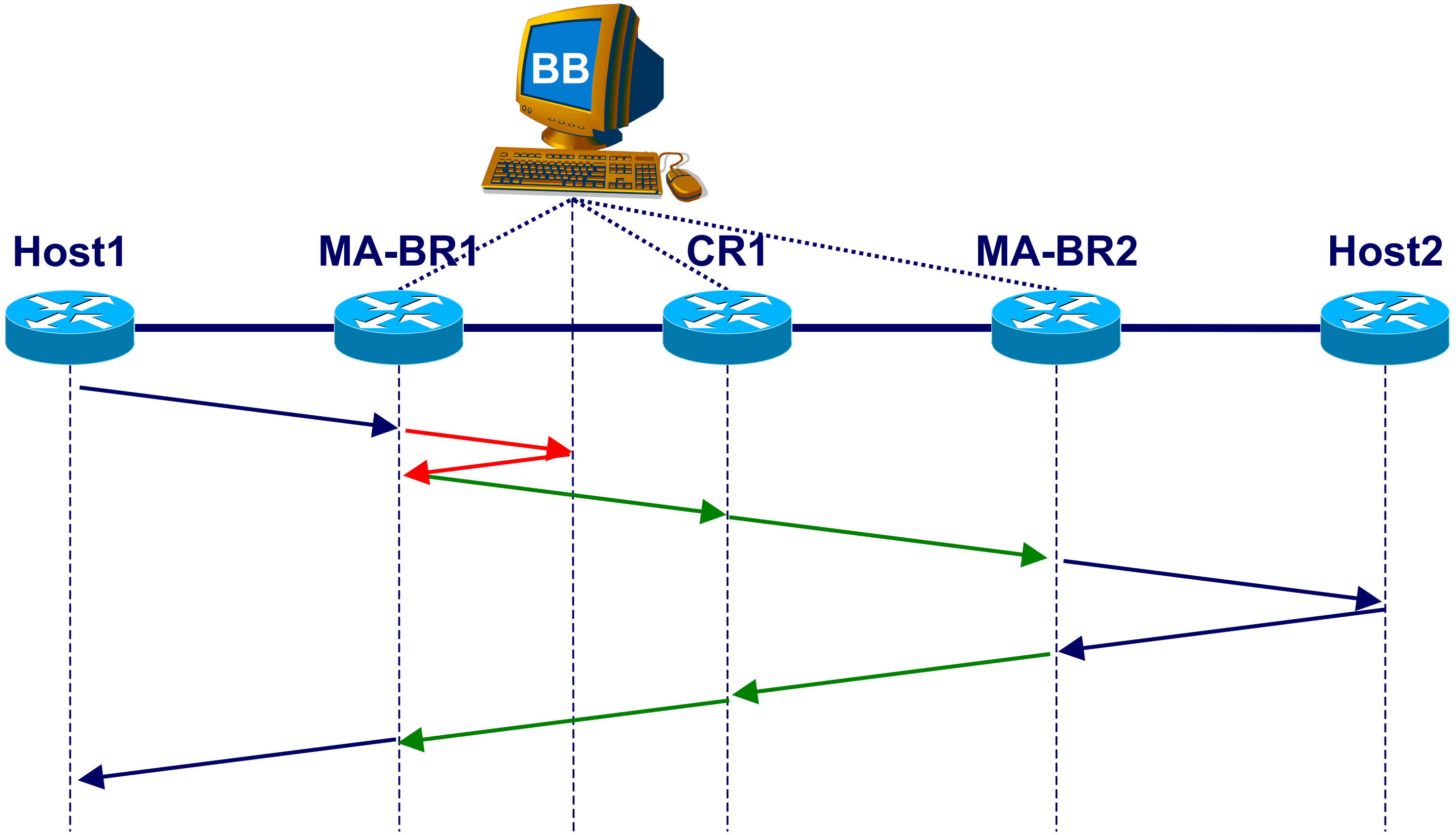
- Solution proposed through the COPS-MAID architecture
  - define a unified and extended COPS semantic, which integrate all the QoS information carried out by the different access protocols
  - translate the different QoS information in a unique format
- This solution transfers the system complexity on the border routers, in which appropriate Inter Working Units (IWUs) are used to map protocol specific messages into generalized client messages.
- A unique COPS client-type (i.e. the COPS-MAID one) can transmit all the information to a unique COPS server, which can be located inside the BB itself.



COPS-MAID extensions

Message Type	Direction	Contents	
Request	PEP → PDP **(MA-BR → BB)**	Traffic originator	- Source host - Ingress MA-BR I/F + Label
		Traffic Terminator	- Destination host - Egress MA-BR I/F
		Traffic description	- resource class/color - setup/ holding lsp priority, - multiple {lsp diffserv type (e-lsp, l-lsp), traffic characterization (RP, LBAP, 3D-LBAP, etc.)}
		QoS description	- Bandwidth, delay - Jitter, loss probability
		LSP recovery behaviour	- Recovery type (path prot., path rest., …) - Diversity type (node, link, SRLG)
		Temporal info	- start time - end time
Decision	PDP → PEP **(BB → MA-BR)**	Label type	- DiffServ, - ATM, - MPLS
		Label	- DSCP, - LSPId
		Explicit Route Object	- primary ERO - backup ERO

Intra–domain signaling

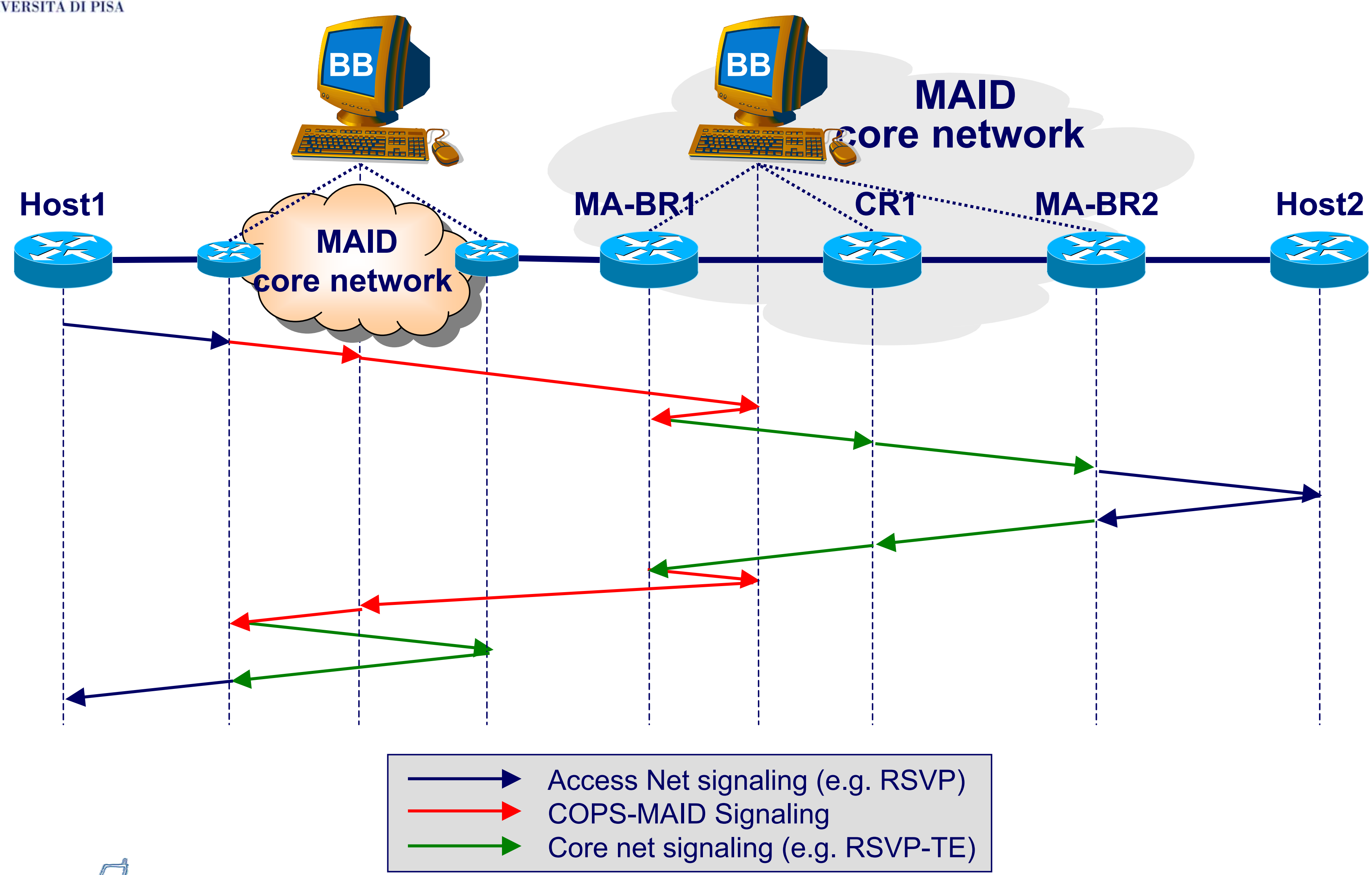


→ Access Net signaling (e.g. RSVP)

→ COPS Signaling

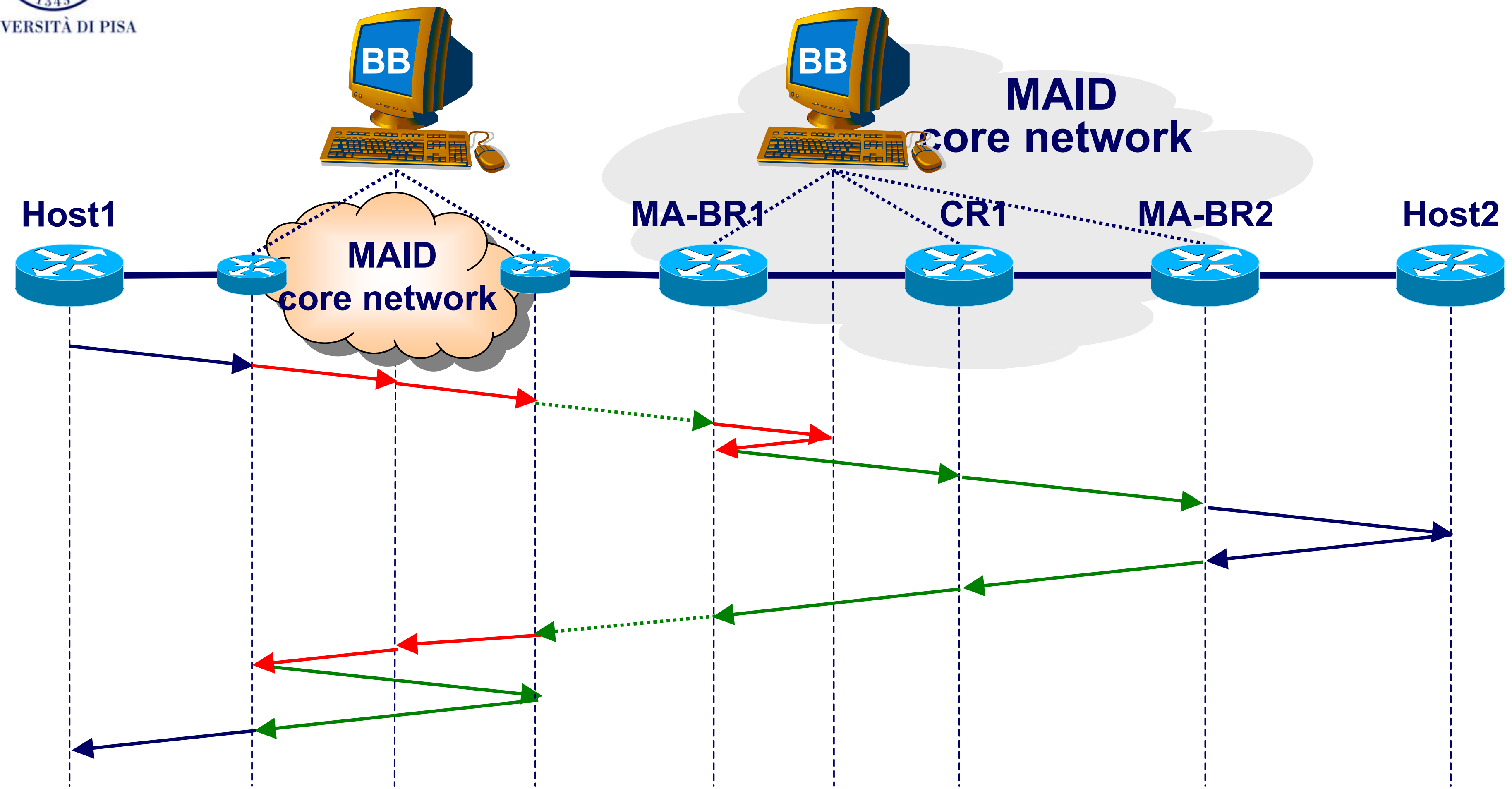
→ Core net signaling (e.g. RSVP-TE)

Inter–domain signaling via COPS-MAID



→ Access Net signaling (e.g. RSVP)
→ COPS-MAID Signaling
→ Core net signaling (e.g. RSVP-TE)

Inter–domain signaling via NNI



- Access Net signaling (e.g. RSVP)
- COPS-MAID Signaling
- Core net signaling (e.g. RSVP-TE)
- NNI signaling (e.g. NNI RSVP-TE)

Outline

- The Multi-protocol Access Inter Domain architecture
  - The Common Open Policy Service and its extensions
  - The seamless inter-domain QoS-IP services

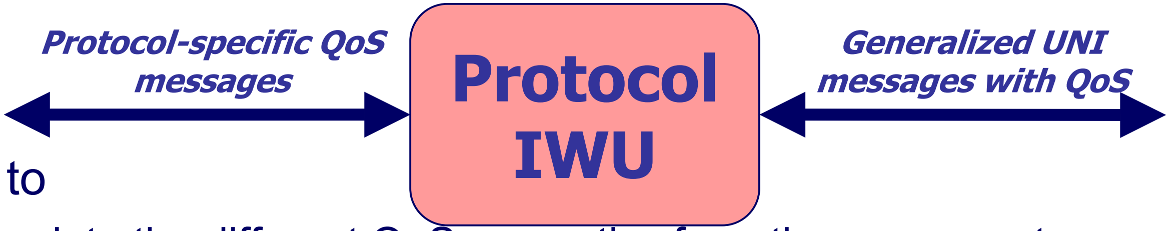
- Inferring QoS-IP parameters from multimedia applications in the MAID architecture
  - The QoS-IP generalized syntax

- Inter-working between the MPEG-4 DMIF and the MAID QoS-IP signaling
  - MPEG-4 architectural overview
  - MPEG-4 DMIF: message flow
  - DMIF IWU implementation

The QoS-IP generalized syntax

- There are 2 basic functionalities to be supported MA-BRs for QoS-IP provisioning across the IP backbone:
  - Understanding of as much as possible QoS semantics from the access net
  - Translating the access QoS into the interior QoS syntax

- … these requirements lead to the implementation of protocol-specific Inter-Working Units (x-IWU)
  - x-IWU acts as on-line translators from the specific-QoS syntax to a generalized QoS-IP syntax (i.e. the UNI syntax !!)


Protocol-specific QoS messages
Protocol IWU
Generalized UNI messages with QoS

- … in order to
  - accommodate the different QoS semantics from the access net
  - be independent of the underlying Data Plane technology

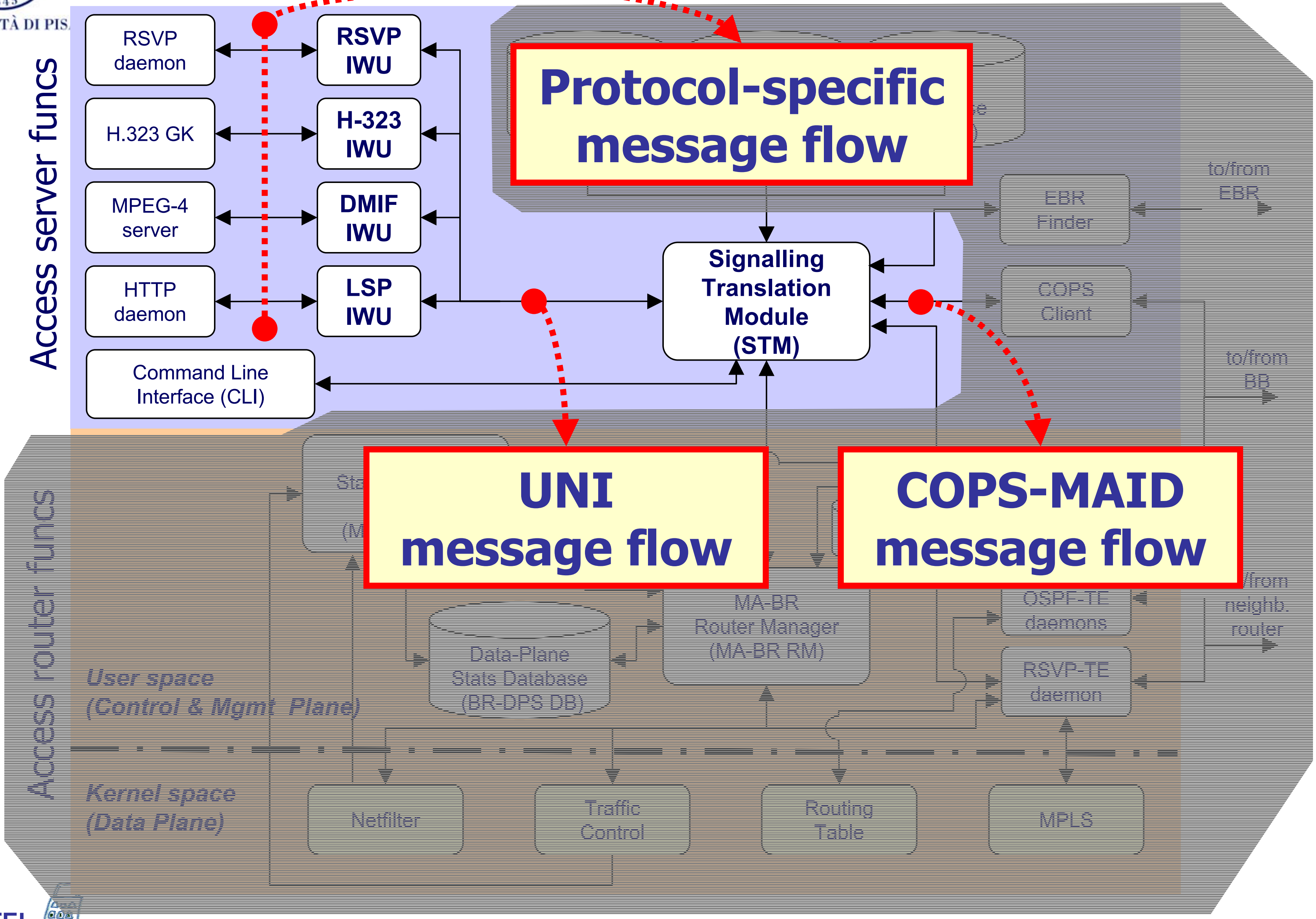
The QoS-IP generalized syntax

- The generalized UNI syntax for setting up a QoS-IP service might be comprised of:

  - **A traffic descriptor**
    - A set of filters to be applied to the involved resources:
      - case INTESERV: src_add, src_port, dst_add, dst_port, protocol (tcp, udp, other)
      - case DIFFSERV: dscp
    - r-b plot, LBAP, 3D-LBAP, CODECS (g711Alaw64k, g711Ulaw64k, etc.)
    - traffic setup/holding priorities
    - resource affinities (Exclude-any/ Include-any/ Include-all color masks)
    - some recovery info (recovery type, diversity type)
    - some temporal info (start_time, end_time)

  - **A QoS descriptor**
    - QoS Class of Service (EF, AF11, AF21, etc.)
    - bandwidth rates (rate_min, rate_max)
    - buffer sizes (size_min, size_max)
    - end-to-end delay (delay_max, delay_min)
    - end-to-end jitter (jitter_max, jitter_min)
    - loss probability (loss_max, loss_min)

The QoS-IP generalized syntax



Protocol-specific message flow

UNI message flow

COPS-MAID message flow

Outline

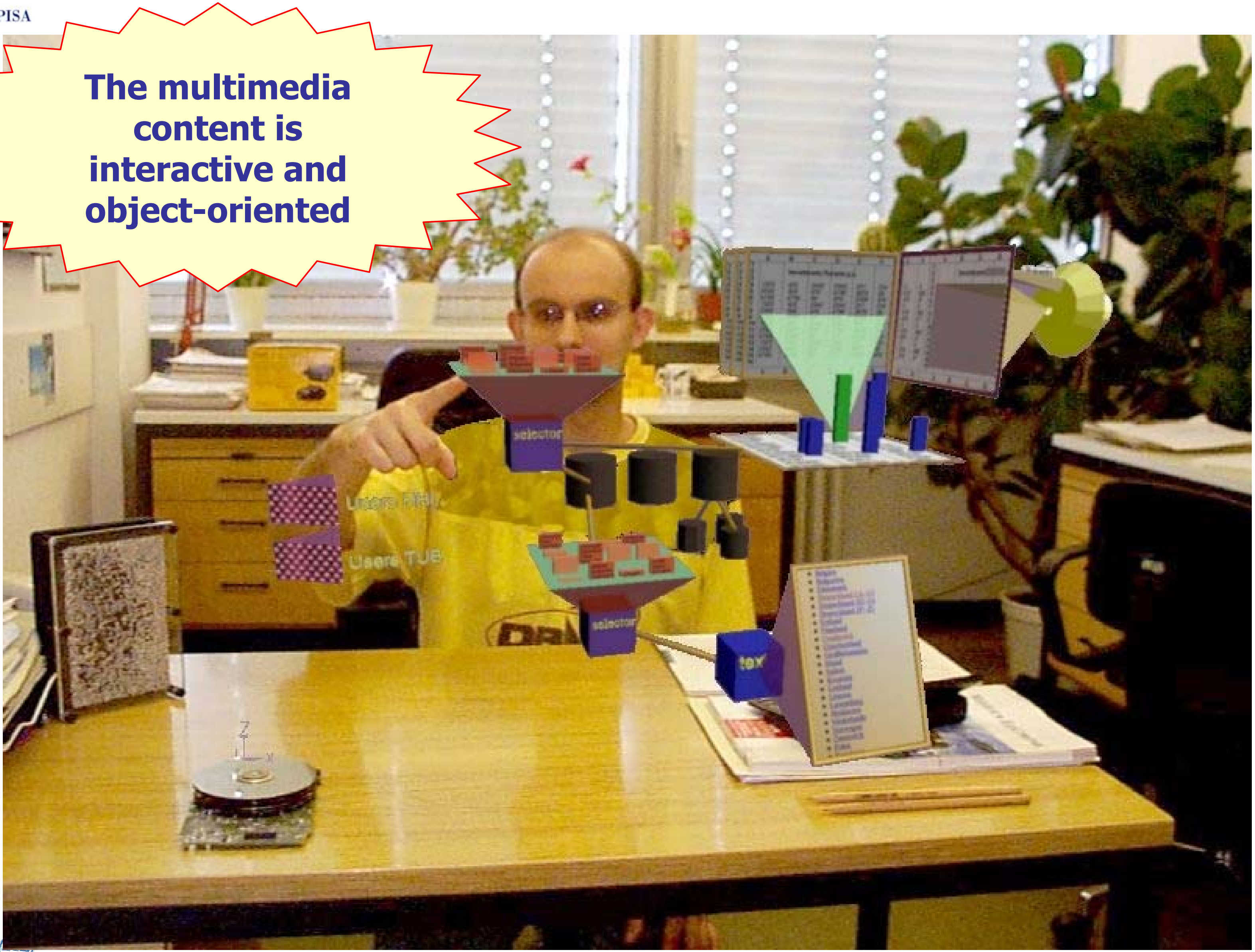
- The Multi-protocol Access Inter Domain architecture
  - The Common Open Policy Service and its extensions
  - The seamless inter-domain QoS-IP services

- Inferring QoS-IP parameters from multimedia applications in the MAID architecture
  - The QoS-IP generalized syntax

- Inter-working between the MPEG-4 DMIF and the MAID QoS-IP signaling
  - MPEG-4 architectural overview
  - MPEG-4 DMIF: message flow
  - DMIF IWU implementation

MPEG-4 architectural overview

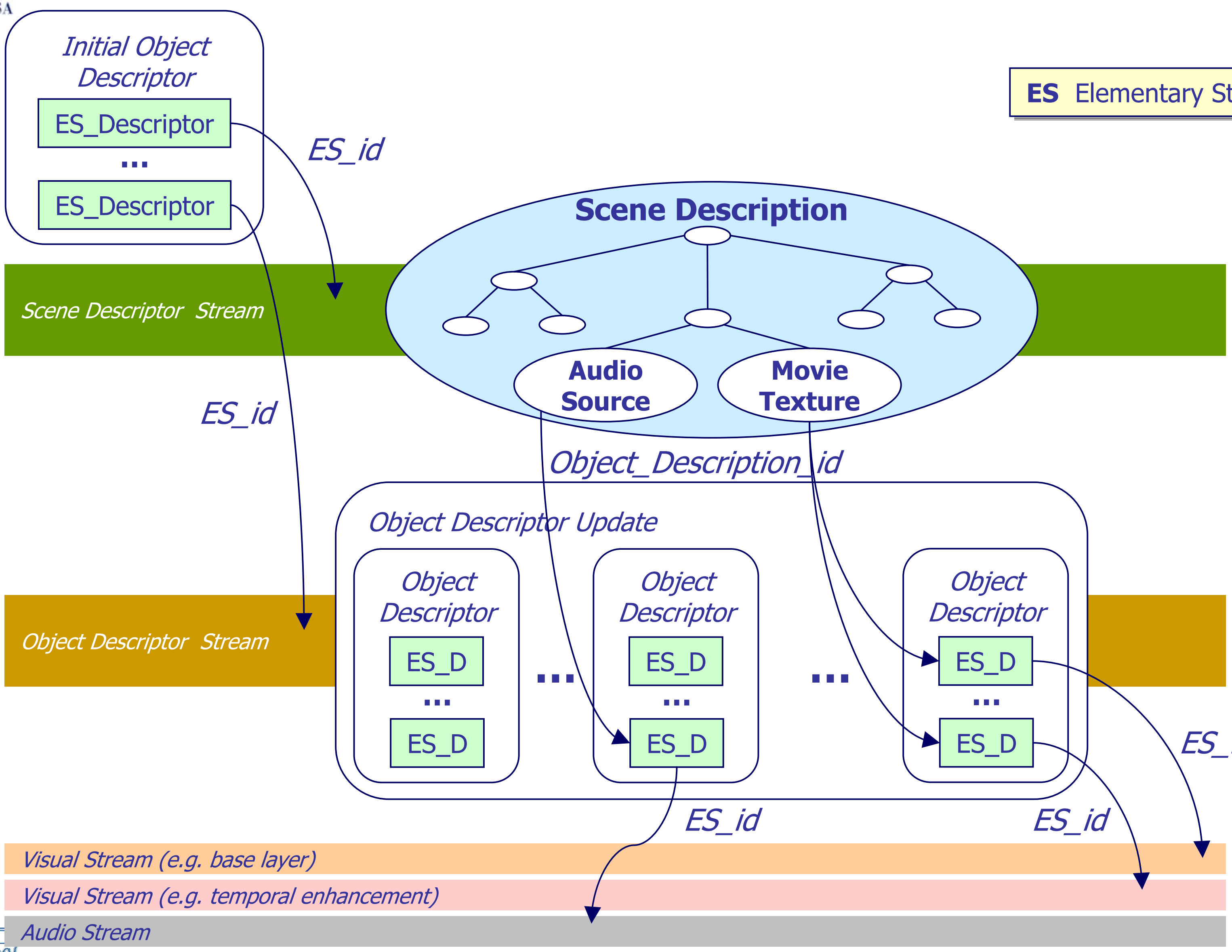
The multimedia content is interactive and object-oriented



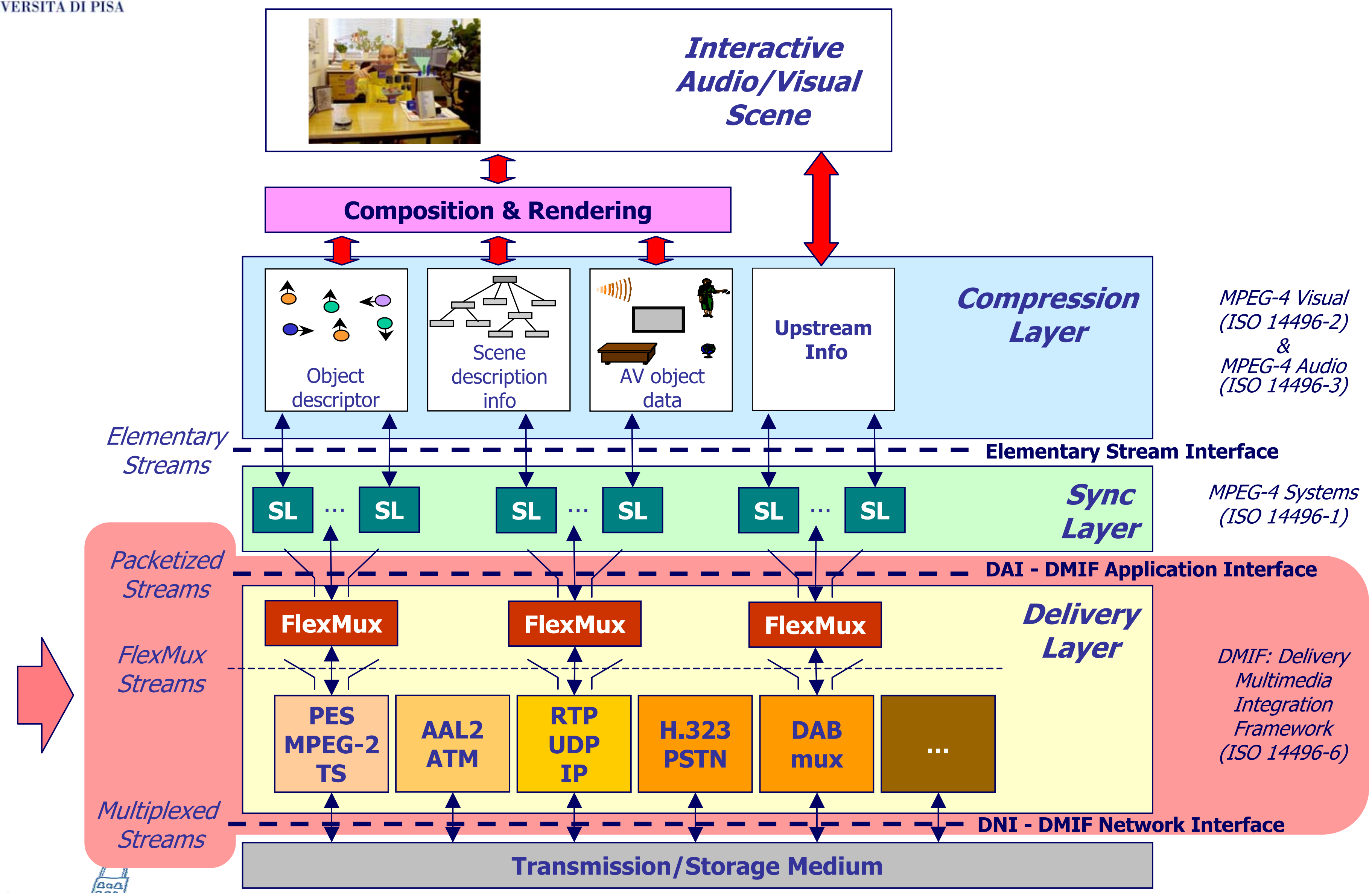
UNIVERSITÀ DI PISA

MPEG-4 architectural overview

ES Elementary Stream



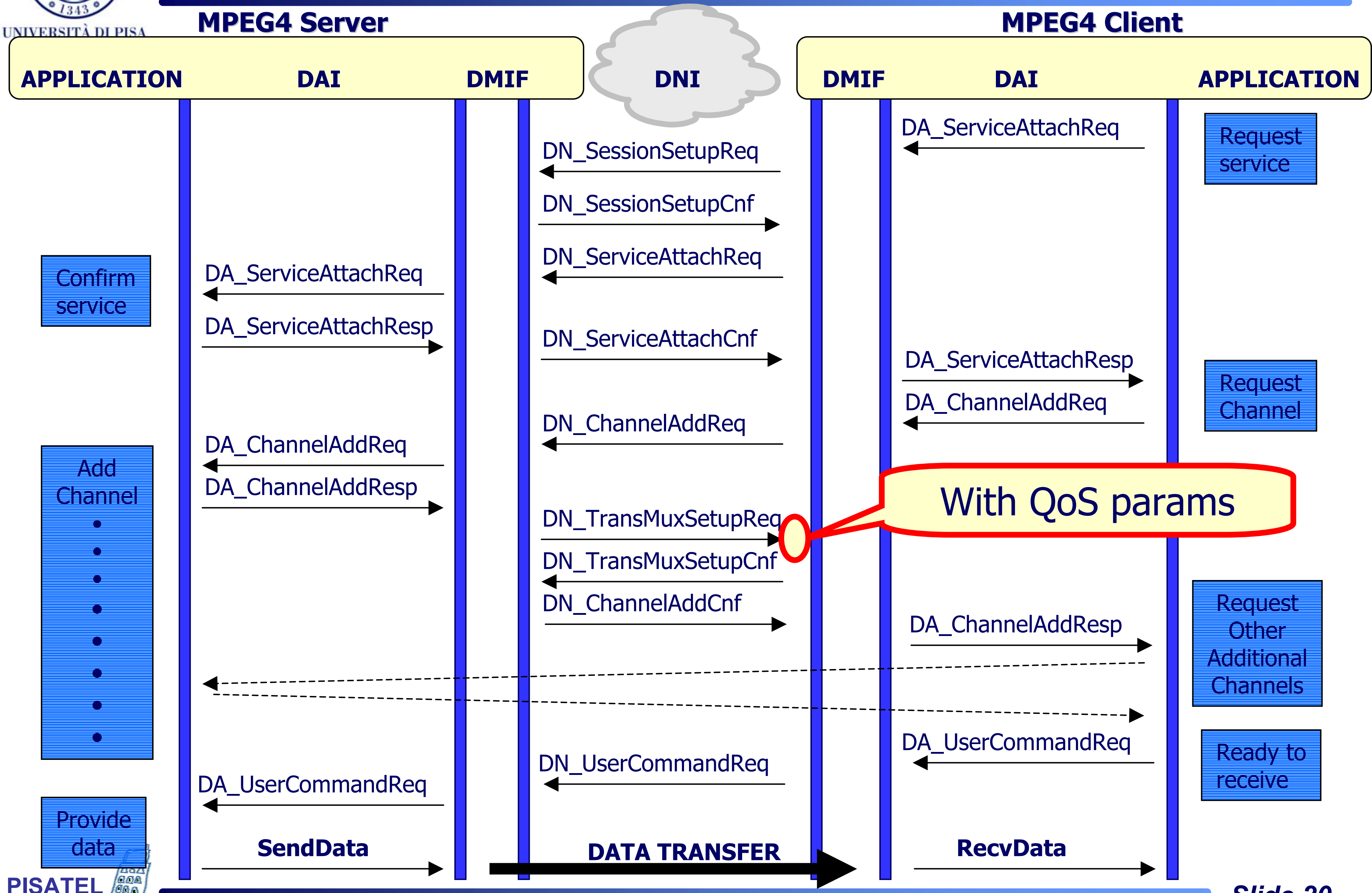
MPEG-4 architectural overview



MPEG-4 architectural overview

- MPEG-4 multimedia contents are based on the compression and delivery of different objects via the DMIF signaling protocol
- The DMIF signaling protocol is a medium-independent session level protocol (somehow equivalent to the FTP)
  - 1st step: opening a session with a peer entity {+ authorization and security checks}
  - 2nd step: delivering of user data
- Each object is delivered on an independent channel with specific parameters describing the multimedia content (Elementary Stream descriptor)

MPEG-4 DMIF: message flow



MPEG-4 DMIF: message flow

- Some of the delivered descriptors are related to the QoS…
- … since the DAI is designed to operate with any kind of multimedia applications, its QoS descriptor is generic.

Metrics	Description
MAX_DELAY	***Maximum delay to be experienced by any PDU***
AVG_DELAY	***Average delay to be experienced by any PDU***
LOSS_PROBABILITY	***Allowable probability of loss of any PDU***
JITTER_TOLERANCE	***Maximum delay variation to be experienced by any PDU***
TRAFFIC_TYPE	***A priority identifier of the stream***
MAX_PDU_SIZE	***Maximum size of a PDU***
AVG_RATE	***Average bit rate measured on a a time window***
MAX_RATE	***Maximum bit rate measured on a a time window***

End-to-end scope

Traffic description

MPEG-4 DMIF: message flow

DN_TransMuxSetupRequest

```
+--------------------------------------------+
|  Network Session ID (10 bytes)             |
+--------------------------------------------+
|  Descriptors Count (1 byte)                |
+--------------------------------------------+
|  Transmux Association Tag - TAT (2 bytes)  |
+--------------------------------------------+
|  Direction (1 byte)                        |
+--------------------------------------------+
|  QoS Descriptors Count (1 byte)            |
+--------------------------------------------+
|  QoS_Tag     (1 byte)                      |
+------------------------------------- ------+
|  QoS_DataLen (1 byte)                      |
+--------------------------------------------+
|  QoS_Data    (variable)                    |
+--------------------------------------------+
//                    ...                    //
+--------------------------------------------+
|  Resources Descriptor Count      (2 bytes) |
+--------------------------------------------+
|  Resources Descriptor Type       (2 bytes) |
+--------------------------------------------+
|  Resources Descriptor Length     (2 bytes) |
+--------------------------------------------+
|  Resources Data Field Count      (2 bytes) |
+--------------------------------------------+
|  Resource Descriptor Data Fieds (variable)|
+--------------------------------------------+
//                    ...                    //
+--------------------------------------------+
//                    ...                    //
+--------------------------------------------+
```

QoS descriptors

Resources

DN_TransMuxSetupConfirm

```
+--------------------------------------------+
|  Descriptors Count (1 byte)                |
+--------------------------------------------+
|  Response (2 byte)                         |
+--------------------------------------------+
|  Resources Descriptor Count     (2 bytes)  |
+--------------------------------------------+
|  Resources Descriptor Type      (2 bytes)  |
+--------------------------------------------+
|  Resources Descriptor Length    (2 bytes)  |
+--------------------------------------------+
|  Resources Data Field Count     (2 bytes)  |
+--------------------------------------------+
|  Resource Descriptor Data Fieds (variable)|
+--------------------------------------------+
|  Resources Descriptor Type      (2 bytes)  |
+--------------------------------------------+
|  Resources Descriptor Length    (2 bytes)  |
+--------------------------------------------+
|  Resources Data Field Count     (2 bytes)  |
+--------------------------------------------+
|  Resource Descriptor Data Fieds (variable)|
+--------------------------------------------+
//                                          //
+--------------------------------------------+
//                  ...                     //
+--------------------------------------------+
```

Resources

e.g. IP resource Descriptor:
- IP_src_addr, IP_src_port
- IP_dst_addr, IP_dst_port
- IP_proto(TCP, UDP, etc.)

DMIF IWU implementation

- The basic action of the DMIF-IWU is to “sniff” the DMIF signaling

- When needed the DMIF-IWU triggers the setup of QoS-IP services across the MAID network
  - The trigger messages are the **DN_TransMuxSetupReq** that may carry the QoS descriptor …

DMIF QoS Descriptor	Generalized UNI	
MAX_DELAY	End-to-end delay	Max delay
AVG_DELAY		Min delay
LOSS_PROBABILITY	Loss probability	
JITTER_TOLERANCE	End-to-end delay jitter	
TRAFFIC_TYPE	LSP setup priority	
MAX_PDU_SIZE	Buffer size	
AVG_RATE	Rate range	Min rate
MAX_RATE		Max rate

Used to determine the QoS class (EF, AF11, etc.)

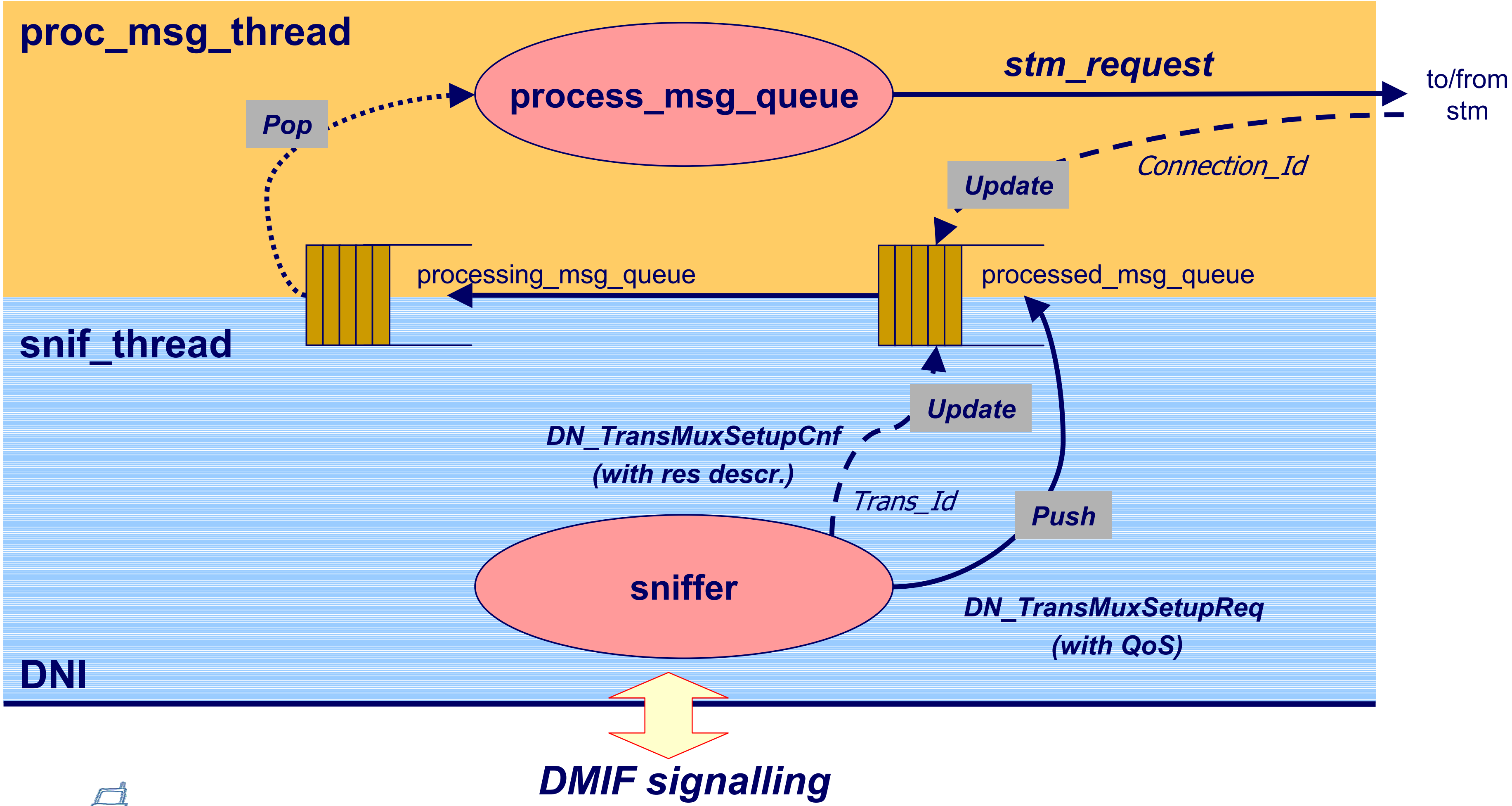
  - … and the **DN_TransMuxSetupCnf** that carries the final resource descriptors (e.g. addresses, ports, protocol)

DMIF IWU implementation

- Two roles are possible for this IWU
  - **Active**: IWU blocks the DMIF channel setup until the Authorization/Admission Control phases are completed on MA-BR (if the requested QoS level cannot be guaranteed the channel is rejected)
  - **Passive:** IWU acts as a passive translator of the DMIF signaling
    - the call signaling flow proceeds unchanged towards the requesting client
    - if the requested QoS level cannot be guaranteed by the MAID network, the channel experiences a Best Effort treatment in the IP network
    - the user will decide if the quality of the received streaming is acceptable or not (→ stop playing)

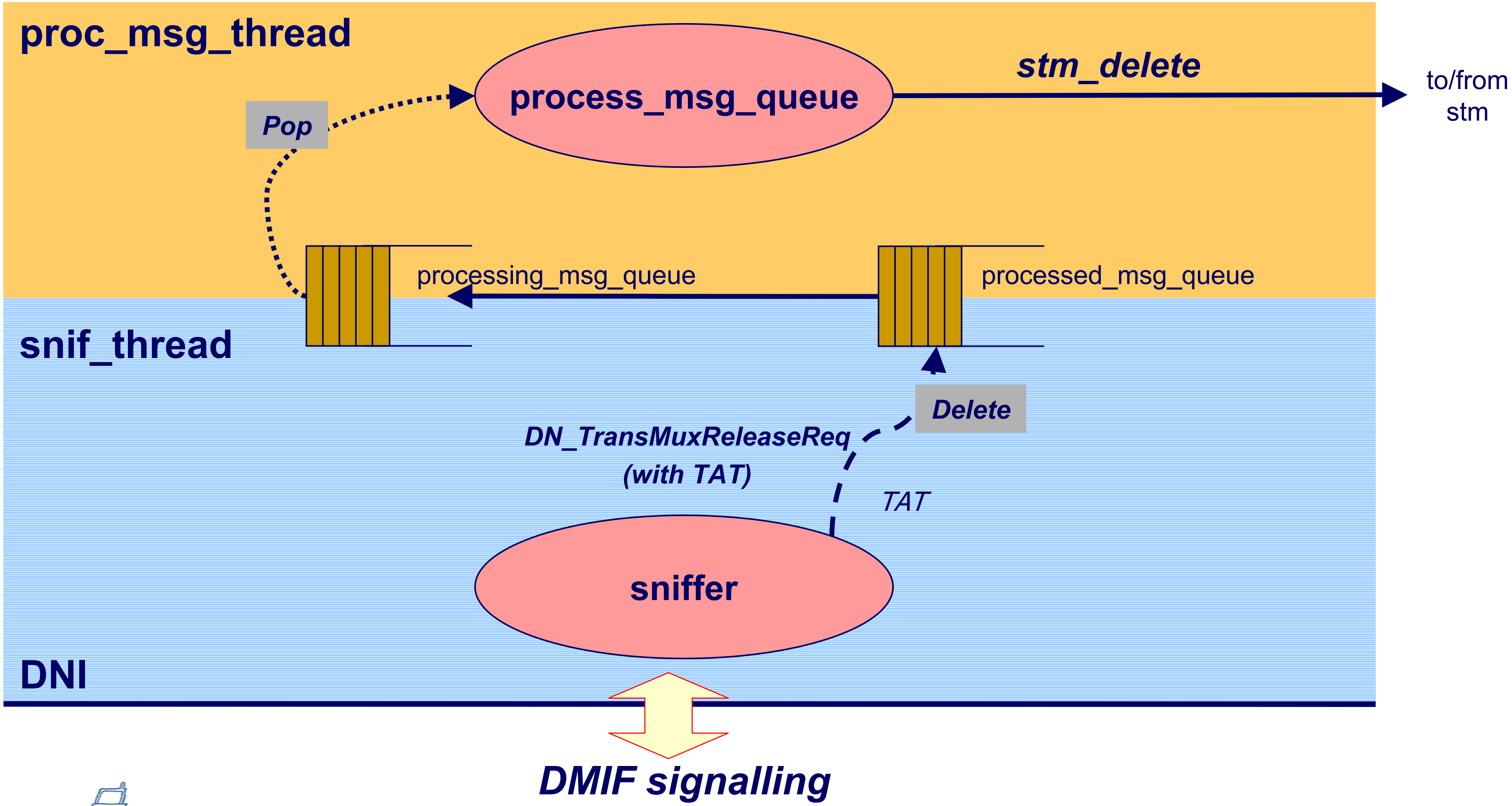
DMIF IWU implementation

- IWU modular/functional decomposition (passive role)

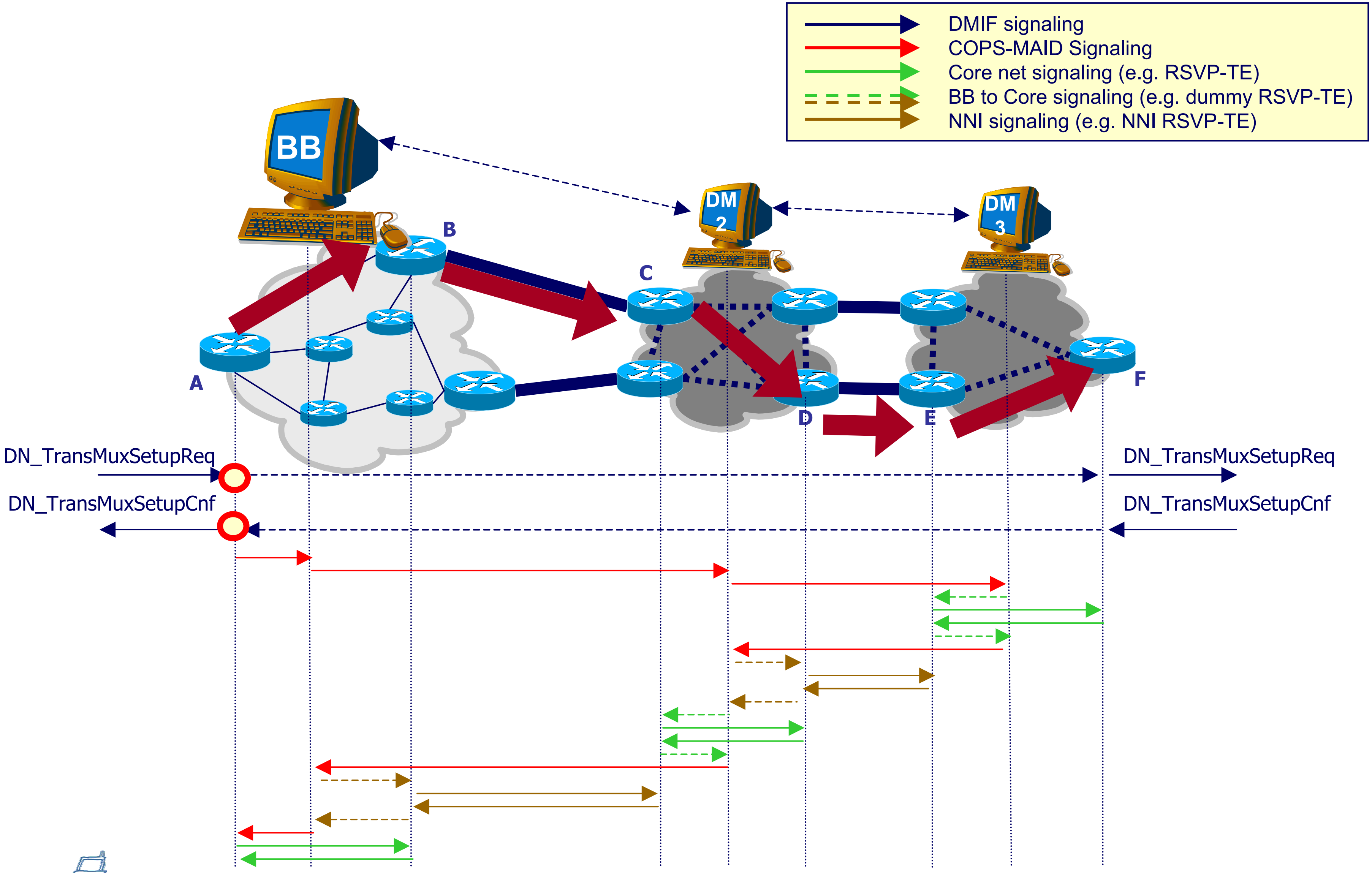


DMIF IWU implementation

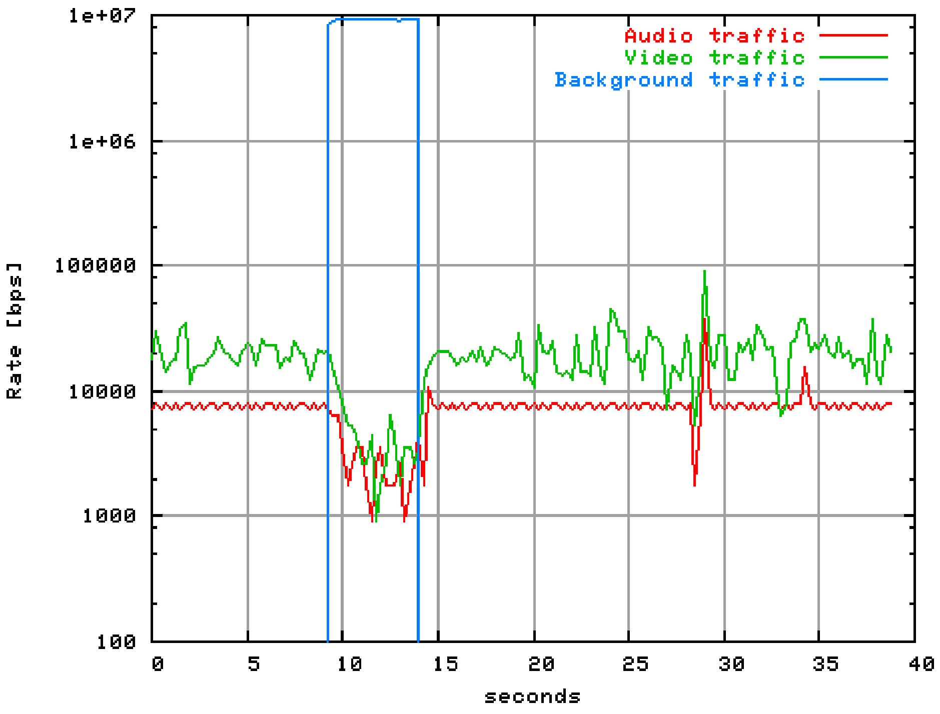
- IWU modular/functional decomposition (passive role)



DMIF-IWU: QoS-IP service setup

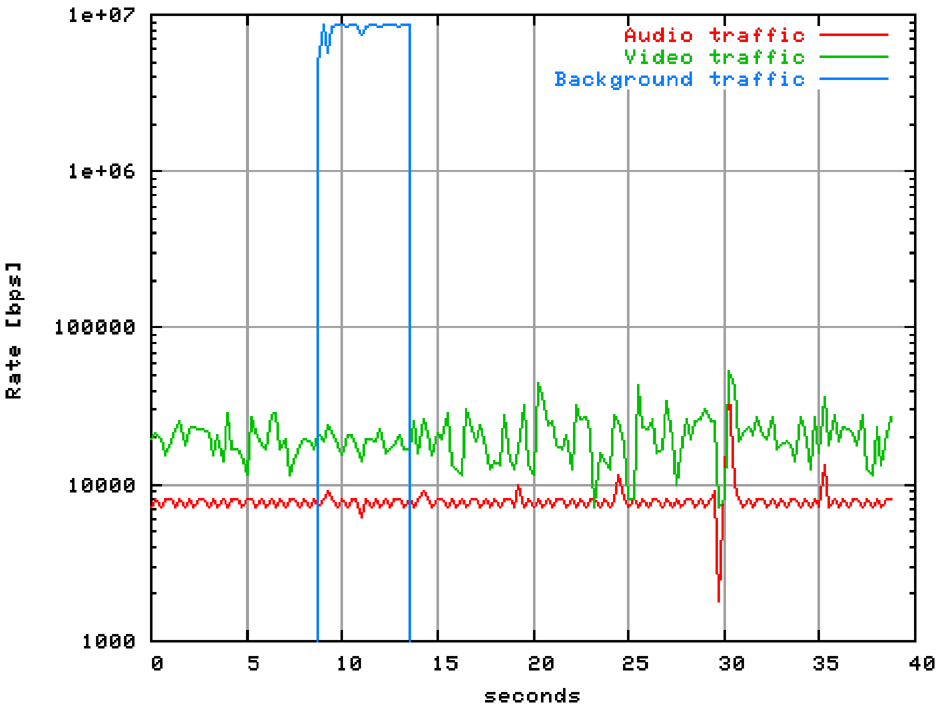


Experimental results



Not differentiated traffic
→ Best Effort treatment

Guaranteed QoS service
→ Protected MPEG-4 streaming


1e+07
1e+06
Rate [bps]
seconds
Audio traffic
Video traffic
Background traffic