ForCES: An implementation

Evangelos Haleplidis (ehalep@ece.upatras.gr) (Speaker)
Odysseas Koufopavlou (odysseas@ece.upatras.gr)
Denazis Spyros (sdena@ece.upatras.gr)
University of Patras
Department of Electrical and Computer Engineer
IETF 70 - Vancouver
December 4, 2007

Presentation Summary

- Driving ForCES
- Protocol/Model Implementation
- Developing ForCES
- Questions

Driving ForCES (1)

- Flexinet FP6-IST1 507646
  - Scalable and modular network architecture offering cross-connect control, switching/routing control, and advanced services management/access functions at the network access points.
- ForCES Implementation consists of:
  - Heartbeats, Association Messages, Simple Configuration Messages, Simple Query Messages. (code based on protocol draft ver06)
  - LFBs:
    - FEProtocolLFB.
    - FEObjectLFB.
    - Incoming LFB.
    - Outgoing LFB.
    - Classifier LFB.

Driving ForCES (2)

- Phosphorus FP6 034115[1]
  - Enhance and demonstrate solutions that facilitate vertical and horizontal communication among applications middleware, existing Network Resource Provisioning Systems, and the proposed Grid-GMPLS Control Plane.
  - UvA[2] & UoP part: Token Based Switch (TBS) is a low-level system for traffic routing at high speeds based on packet authentication.
    - UvA partners: Mihai Cristea[3], Yuri Demchenko[4].
  - Code based on Protocol draft-11.

[1] http://www.ist-phosphorus.eu/

[2] University of Amsterdam

[3] cristea@science.uva.nl

[4] demch@science.uva.nl

Protocol/Model Implementation (1)

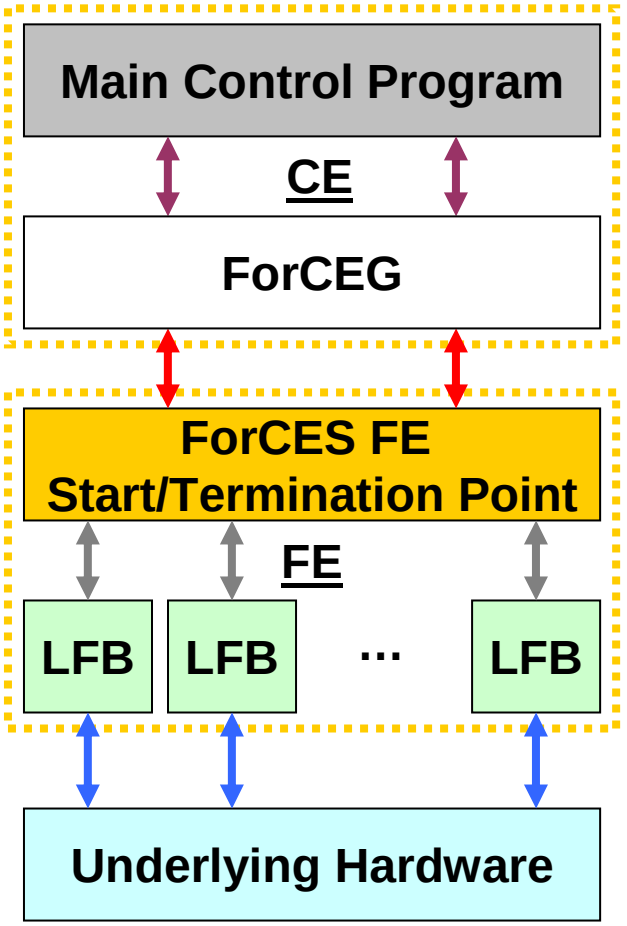
- Current protocol implementation consists of:
  - 1 CE (ForCEG: ForCES Gateway)
  - 1 FE containing:
    - FEProtocolLFB.
    - FEObjectLFB.
    - Rx LFB.
    - Tx LFB.
    - Token Switch LFB.
    - Token Builder LFB.
- TML:
  - TCP/IP.

Protocol/Model Implementation (2)

- Protocol Parts implemented as yet:
  - Association messages.
  - Heartbeat messages.
  - SET messages.
  - GET messages.

Protocol/Model Implementation (3)

- CE (ForCEG):
  - Basic CE functionality.
  - Incorporates a Web Service for sending commands.
  - Commands are processed in XML and translated in ForCES.

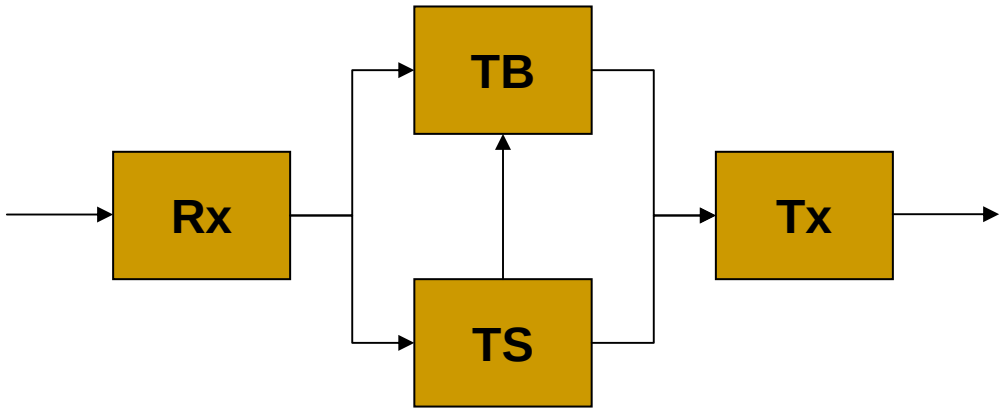


Protocol/Model Implementation (4)

- FE:
  - Intel IXP enp2611(w/o security) / 2850(w security).
    - Mihai Cristea (cristea@science.uva.nl) (TBS development & IXP coding)
  - Token Based Switch
    - Low-level system for traffic routing at high speeds based on packet authentication.
    - Two major entities:
      - Token Builder (TB): Computes a token and insert it into the packet.
      - Token Switch (TS):
        - Receives a packet with a token.
        - Computes a local token (as TB)
        - Compares tokens. (if match the packet is authorised)

Protocol/Model Implementation (5)

- TBS:
  - Mapping of TB and TS varies on the application type.
    - Entrance point of a packet: Only TB.
    - Entrance point to an authorized network: Only TS.
    - Internal change of authorized network (need to prepare a new token): TS & TB.
  - One configurable FE.



Protocol/Model Implementation (5)

- Rx Model:

dataTypeDefs

dataTypeDef (2)

	{} name	{} synopsis	{} atomic	{} struct
1	Ports	Values that can be applied to Incoming and Outoging Port for Condition Forwarding	**atomic** {} base... u8 **rangeRestriction** **allowedRange** = max 0 = min 9	
2	ConditionForwarding	A Condition Forwarding Case. Has incoming port and outgoing port		**struct** **component** (3)

component (3)

	= c	{} name	{} synopsis	{} typeRef
1	1	IncomingPort	The Incoming Port/MicroEngine	Ports
2	2	Condition	The condition upon which from 1 will go to 2	u8
3	3	OutgoingPort	The Outgoing Port/MicroEngine	Ports

LFBClassDefs

LFBClassDef

= LFBClassID	3
{} name	RxLFB
{} synopsis	The port Rx LFB for the FE - TVS
{} version	1.0

components

component (2)

	= c	= acce...	{} name	{} synopsis	{} typeRef
1	1	read-write	Condition_Fwd	An array with condition forwarding	ConditionForwarding
2	2	read-reset	Packet_Count	The number of packets that are coming into the IXP	uint32

Protocol/Model Implementation (6)

- TB Model:

dataTypeDefs

dataTypeDef (5)

	name	synopsis	Comment	struct	array	atomic
1	Tuple	A Tuple from the TVS Ticket	LRI, TokenKey, NewLRI, NewTokenKey, Port1, Port2, IPPacketMask, IPSource, IPDestination, PortSource, PortDestination, Status	struct		
2	ActiveTuples	An Array with the Active Tuples			array: type = variable-size; typeRef = ActiveTupleCount	
3	Ports	Values that can be applied to Incoming and Outoging Port for Condition Forwarding				atomic
4	ConditionForwarding	A Condition Forwarding Case. Has incoming port and outgoing port		struct		
5	ActiveTupleCount	An Active Tuples with it's counters		struct		

LFBClassDefs

LFBClassDef

LFBClassID	5
name	TBLfb
synopsis	Token Builder LFB
version	1.0

components

component (3)

	componentID	acce...	name	synopsis	typeRef
1	1	read-write	Tuples	The tuples inside the TB	ActiveTuples
2	2	read-write	Condition_Fwd	An array with condition forwarding	ConditionForwarding
3	3	read-reset	Packet_Count	The number of packets that are coming into the IXP	uint32

Protocol/Model Implementation (7)

- TS Model:

dataTypeDefs

dataTypeDef (5)

	name	synopsis	Comment	struct	array	atomic
1	Tuple	A Tuple from the TVS Ticket	LRI, TokenKey, NewLRI, NewTokenKey, Port1, Port2, IPPacketMask, IPSource, IPDestination, PortSource, PortDestination, Status	struct		
2	ActiveTuples	An Array with the Active Tuples and their counters			array typ...	
3	ActiveTupleCount	An Active Tuples with it's counters		struct		
4	Ports	Values that can be applied to Incoming and Outoging Port for Condition Forwarding				atomic
5	ConditionForwarding	A Condition Forwarding Case. Has incoming port and outgoing port		struct		

LFBClassDefs

LFBClassDef

LFBClassID	6
name	TSLfb
synopsis	Token Switch LFB
version	1.0

components

component (2)

	componentID	access	name	synopsis	typeRef
1	1	read-write	Tuples	The tuples inside the TB	ActiveTuples
2	2	read-write	Condition_Fwd	An array with condition forwarding	ConditionForwarding

events

baseID	1

event

eventID	1
name	BadCountSurpassed
synopsis	If the Bad Count is surpassed the LFB will send a message to the CE

eventTarget

eventField	Tuples
eventSubscript	_TupleEntry_
eventField	BadTokenCount

eventCreated

eventGreaterT...	255

eventReports

eventReport

eventFi...	Tuples
eventSu...	_TupleEntry_
eventField (2)	

Protocol/Model Implementation (8)

- Tx Model:

- dataTypeDefs
  - dataTypeDef
    - {} name: Ports
    - {} synopsis: Values that can be applied to Incoming and Outoging Port for Condition Forwarding
    - atomic
      - {} baseType: u8
      - rangeRestriction
        - allowedRange
          - = max: 0
          - = min: 9
- LFBClassDefs
  - LFBClassDef
    - = LFBClassID: 4
    - {} name: TxLFB
    - {} synopsis: The port Tx LFB for the FE - TVS
    - {} version: 1.0
    - components
      - component (2)

	= componentID	= access	{} name	{} synopsis	{} typeRef
1	1	read-only	OutputPort	The output port of IXP	Ports
2	2	read-reset	Packet_Count	The number of packets that are leaving the IXP	uint32

Developing ForCES (1)

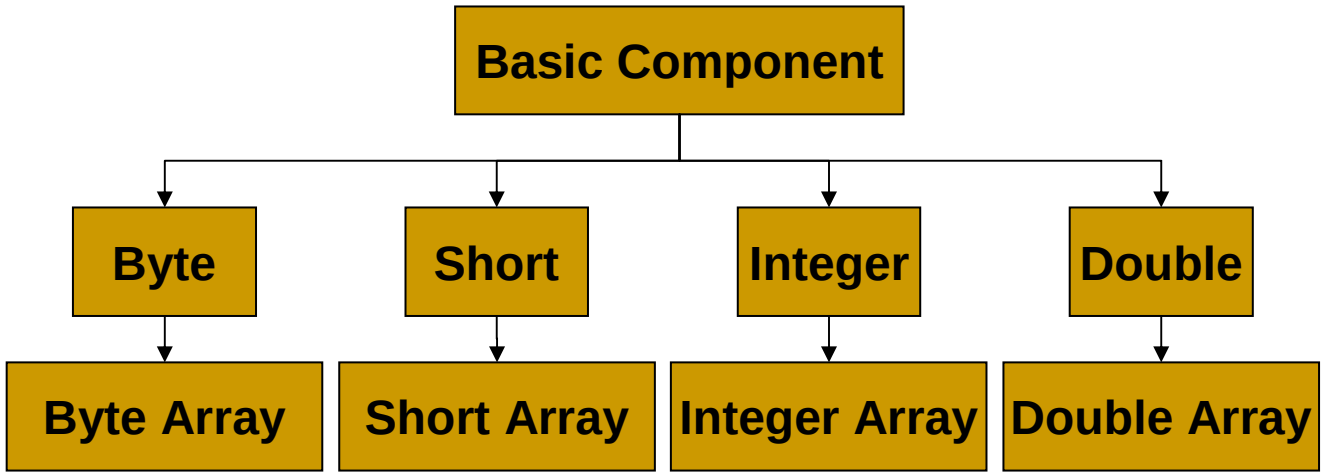
- Challenges encountered:
  - Hardware not ForCES compatible.
  - Complex Model Components.
  - Dynamic Protocol Messages.
  - Protocol Interface.

Developing ForCES (2)

- FE
  - Java (ver6)
    - Pro's:
      - Easier to code and handle.
      - No use of pointers.
    - Con's
      - Different variable types. (e.g. There are no unsigned types!)
      - No system calls for use with hardware.
  - C++ Code (ver11)
    - Pro's
      - System calls.
    - Con's
      - Pointer usage may cause problems.

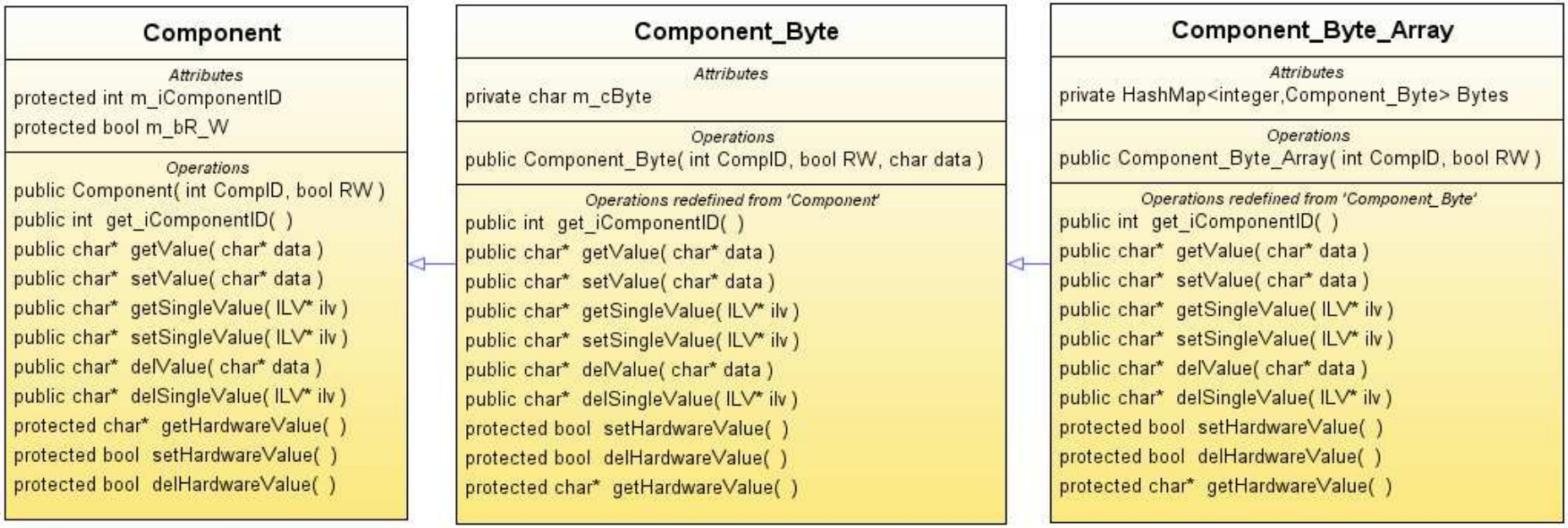
Developing ForCES (3)

- FE:
  - Each Component has a function that sets/gets/dels actual hardware values.
  - Easy to create LFB Components based on hierarchy.



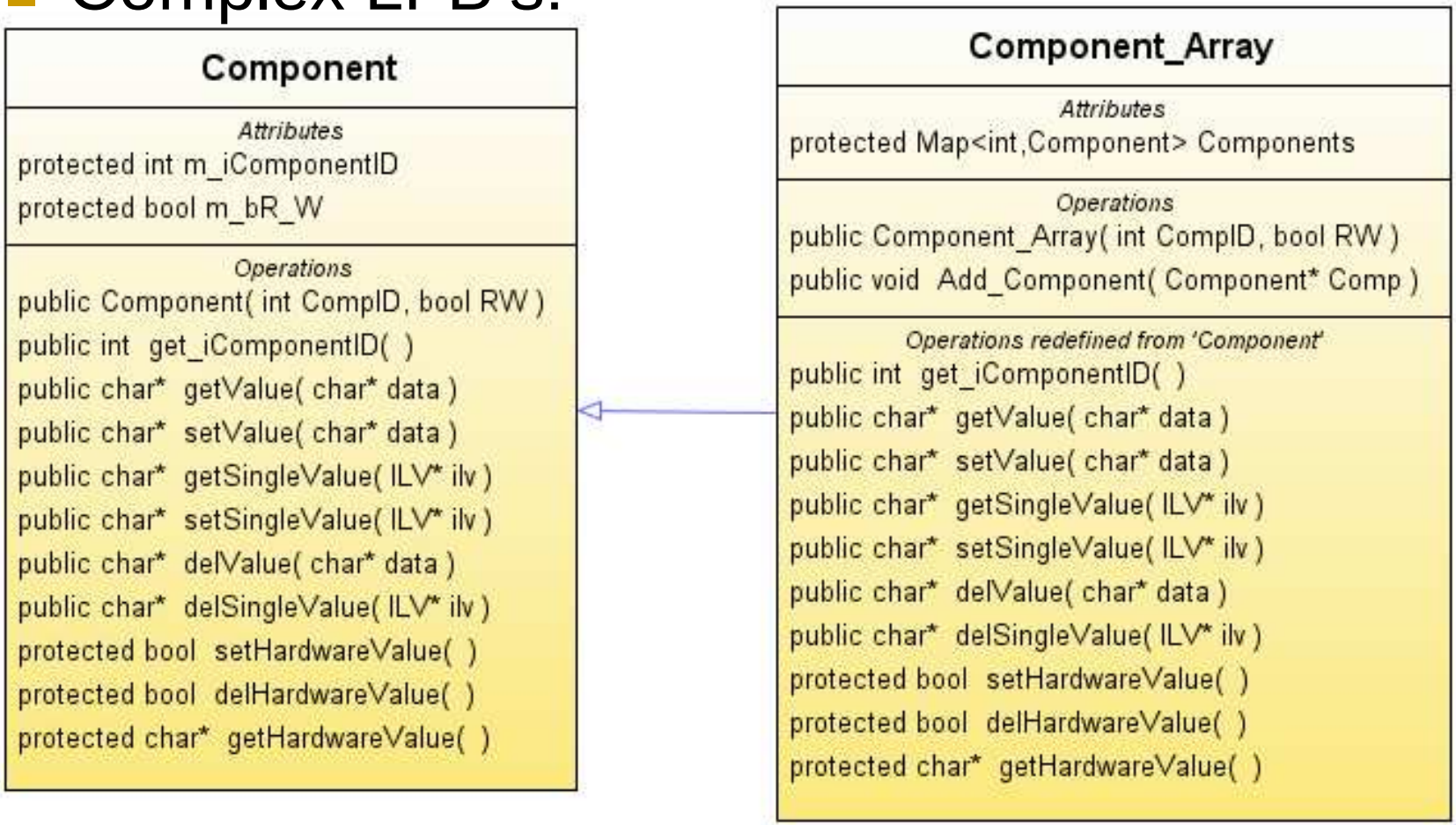
Developing ForCES (4)

- LFB Component Hierarchy:



Developing ForCES (5)

- Complex LFB's:



Developing ForCES (6)

- Protocol Basic Messages

Basic_TLV

Attributes

protected short m_shType
protected short m_shLength
protected char* m_pData
protected List<Basic_TLV> m_lNextTLV

Operations

public Basic_TLV( )
public Basic_TLV(char* raw_tlv)
public Basic_TLV(short Type, short Length, char* Data)
public short get_shType( )
public short get_shLength( )
public char* get_pData( )
public void addTLV( )
public Basic_TLV* get_NextTLV( )
public void replace_previousTLV(Basic_TLV* tlv)
public int Calculate_TLVs_length( )
public char* serialize_me( )

ILV

Attributes

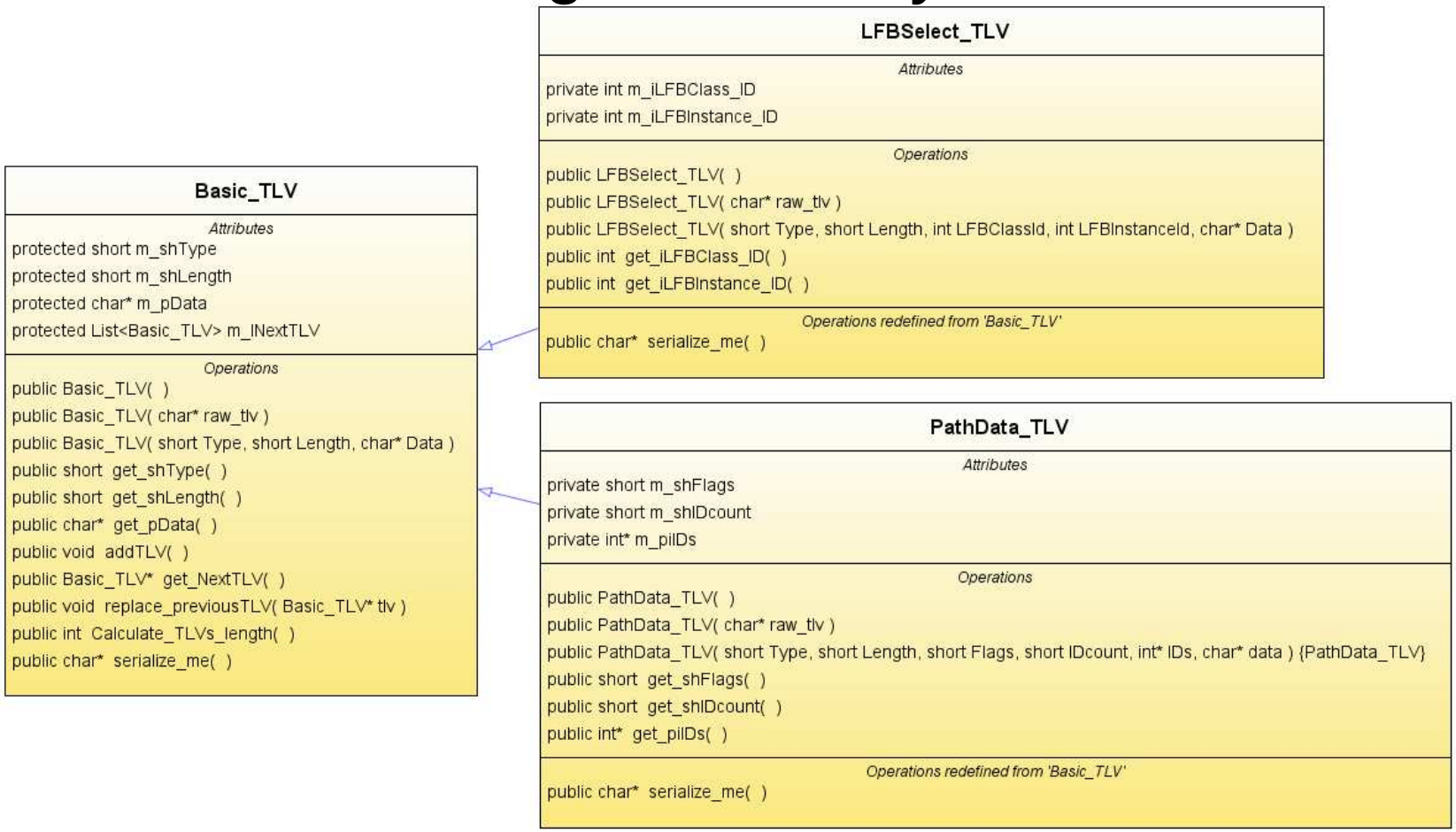
private int m_iIndex
private int m_iLength
private char* m_pData

Operations

public ILV( )
public ILV(char* raw_ilv)
public ILV(int Index, int Length, char* Data)
public int get_iIndex( )
public int get_iLength( )
public char* get_pData( )
public char* serialize_me( )

Developing ForCES (7)

- Protocol Message Hierarchy

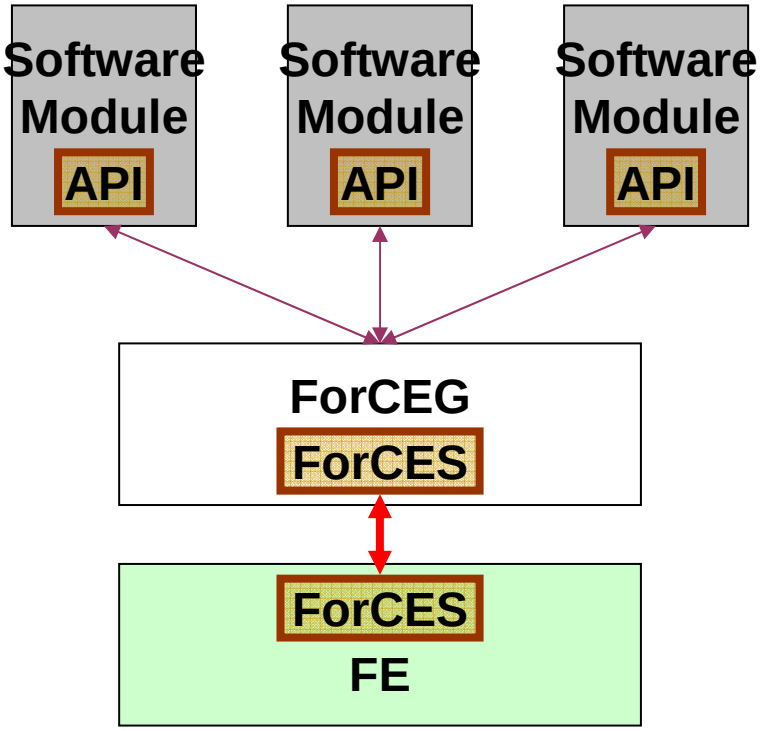


Developing ForCES (8)

- ForCES is a protocol for configuration of the Forwarding Plane.
- What happens when multiple programs need to configure the same FE? Who controls what?
- Multiple CE controlling an FE or a single manageable CE controlling an FE?
- Need for an Open API between CE and programs for issuing commands.

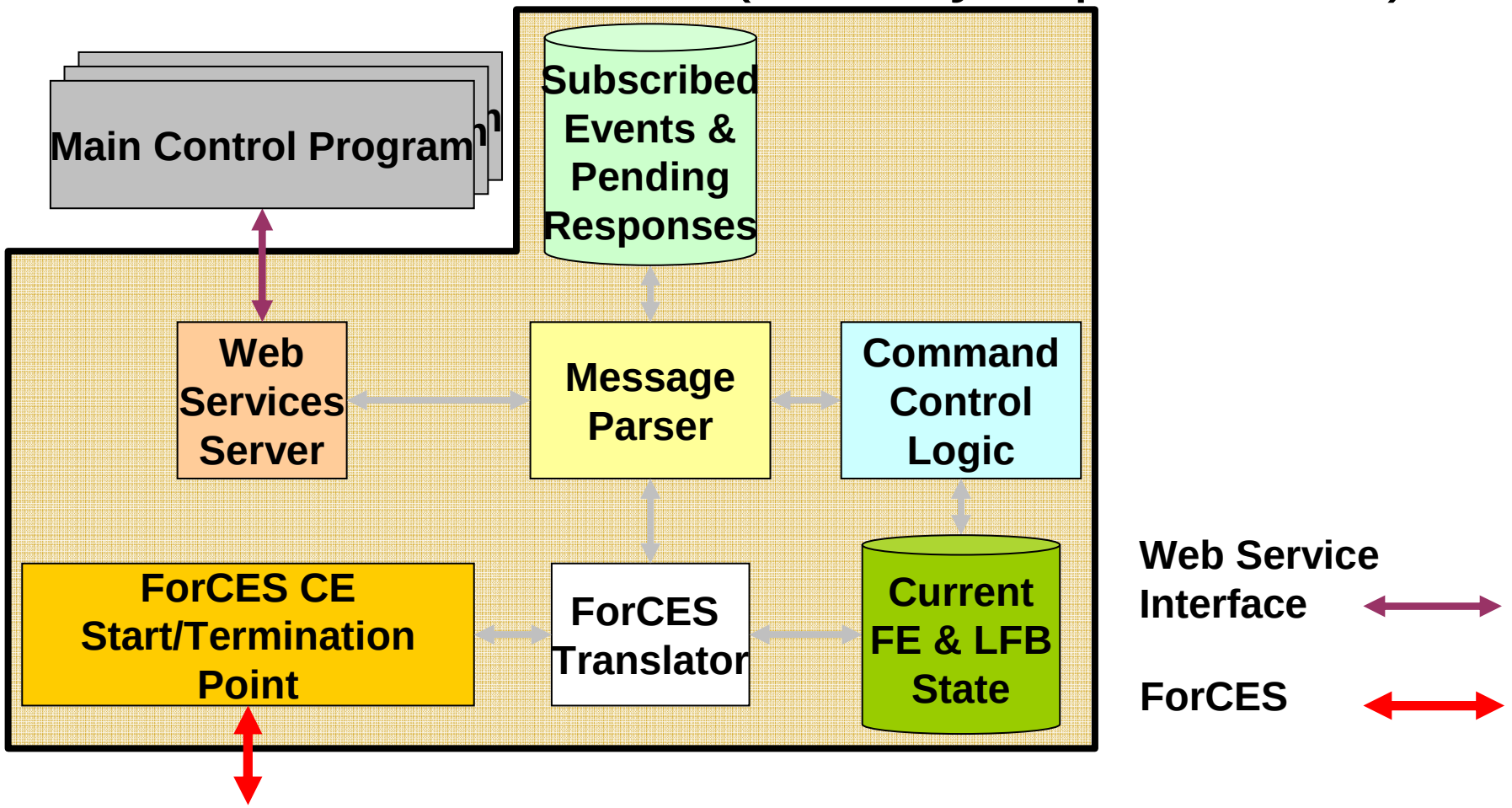
Developing ForCES (9)

- Translates commands from a Generic Web Service API into ForCES packets.
- Conceal ForCES model & protocol from programs.
- Connections of multiple programs into one Forwarding Element.
- Advertise API.



Developing ForCES (10)

- ForCEG Architecture (not fully implemented)



Questions (Ours)

- How will ForCES be used by higher layer applications & to alleviate network services?
- What is the relationship between ForCES and Netconf? Similarities / Differences?

Questions (Yours)

- Any comments are welcome

Thank you!