

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 034115¹
 - 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.
 - UvA² & UoP part: Token Based Switch (TBS) is a lowlevel system for traffic routing at high speeds based on packet authentication.
 - UvA partners: Mihai Cristea³, Yuri Demchenko⁴.
 - Code based on Protocol draft-11.

Protocol/Model Implementation (1) \blacksquare



- 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.

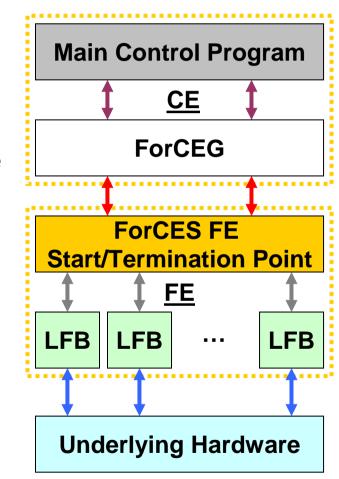
Heartbeat messages.

□ 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)



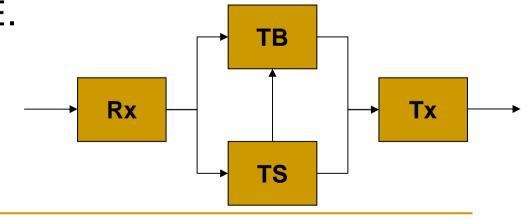
- Intel IXP enp2611(w/o security) / 2850(w security).
 - Mihai Cristea (cristea@science.uva.nl) (TBS development & IXP coding)
- Token Based Switch
 - Low-level system for traffic routing at high speeds based on packet authentication.
 - Two major entities:
 - Token Builder (TB): Computes a token and 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.





Rx Model:

| ntaTypeDef (2) | 10 102111 102 | 1726 | | | | 14580 | | | | |
|-----------------|---|--|---------------|---|---------------------|---------|-------------------|--------------|--|------------|
| () name | () synopsis | () atomic | | | | () stru | ct | | | |
| Ports | Values that can be applied to Incoming and Outoging Port for Condition | ▲ atomic | | | | | | | | |
| | Forwarding | {) base | 118 | | | - | | | | |
| | | and and a state of the state of | testriction | | | | | | | |
| | | Tanger | A allowedRa | nde | | | | | | |
| | | | | philip | 0 | | | | | |
| | | | 202 | and the second | 9 | | | | | |
| ConditionForwar | ding A Condition | And I have | | Plan - Andre | 8 | 🔺 stru | ct | | | |
| | Forwarding Case. Has incoming port and outgoing port | | | | | 1 | | ponent (3) | | |
| | | | | | | - | the second second | c () name | () synopsis | () typeRef |
| | | | | | | | 11 | | a second address of the second s | Ports |
| | | | | | | | 22 | Condition | Contract and the contract of t | u8 |
| nssDefs | | | | | | | 33 | OutgoingPort | – The Outgoing PortMicroEngine | Ports |
| BClassDef | | | | | | | | | | |
| = LFBClassID | 3 | | | | | | | | | |
| () name | R×LFB | | | | | | | | | |
| () synopsis | The port Rx LFB for | the FE - TVS | | | | | | | | |
| () version | 1.0 | anerica desta | | | | | | | | |
| - component | IS | A1873 | | | | | | | | |
| | A component (2) | | | | | | | | | |
| | | = (= acce | () name | <pre>() synopsis</pre> | () 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 | uint32 | | | | | |



| | 🔺 dataTypeD | ef (| 5) | | | | | | | |
|-------------|-------------------------------|-------|--------------------------------|--|--|------------|---|--|---------------------|---------|
| | | | () name | () synopsis | (Comment | () struct | () array | | | () atom |
| | | 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 | <i>•</i> | | 🔺 array | 42 | | |
| | | | | | | | | = type | variable-size | 1 |
| | - | | | | 2 | | _ | () typeRef | ActiveTupleCount | |
| | | 3 | Ports | Values that can be applied to Incoming and Outoging Port for Condition Forwarding | | | | | | ≝ atom |
| | | 4 | | A Condition Forwarding Case. Has incoming port and outgoing port | | ≝ struct | | | | |
| | | 5 | | An Active Tuples with it's counters | | 💌 struct | | | | |
| FBClassDefs | | | | | | | | | | |
| | LFBClassD | 193 y | | | | | | | | |
| | | - 14 | | 5 | | | | | | |
| | | | | TBLfb | | | | | | |
| | | | | Token Builder LFB | | | | | | |
| | | | () version | 1.0 | | | | | | |
| | | | components | | | | | | | |
| | | | -1 | component (3) | | | 1.22 | 1.55 | 192 | 1 |
| | | | | | = componentID | = acce | 1.55 | synopsis | () typeRef | |
| | | | | 1 | S | read-write | Tuples | The tuples inside the TB | ActiveTuples | |
| | | | | 2 | 2 | read-write | Condition_Fwd | An array with condition forwarding | ConditionForwarding | 3 |
| | | | | 3 | 3 | read-reset | 1994-1996-1996-1996-1997-1996-1997-1996-1997-1996-1997-1996-1997-1996-1997-1996-1997-1996-1997-1996- 1996-1996-1996-1996-1996-1996-199 | The number of packets that are coming into the IXP | uint32 | |

TR Model



TS Model:

| dataTypeDef (5) | | 0.00 | | | | | |
|--------------------------------|--|--|---|--------------------|-------------------------------|----------------------|--|
| () 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 | 4 | | |
| 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 ConditionForwardin | g A Condition Forwarding Case. Has incoming port and outgoing port | | 🖈 struct | | | | |
| ClassDefs | | | | | -01 | | |
| 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 | |
| 100 | 2 | 2 | read-write | Condition Fwo | An array with condition forw | arding ConditionForw | |
| ▲ events | | | | | | | |
| | = baseID | 1 | | | | | |
| | A event | | | | | | |
| | | = eventID | 1 | | | | |
| | | () name | BadCountSurpassed | | | | |
| | | () synopsis | and the state of the | rpassed the LFB | will send a message to the CE | | |
| | | ▲ eventTarget | | | | | |
| | | | () eventField | Tuples | | | |
| | | | () eventSubscript | ener Gibbon Marana | | | |
| | | 122 | () eventField | BadTokenCour | h. | | |
| | | eventCreated | | | | | |
| | | oromoroacou | () eventGreaterT | 255 | | | |
| | | ■ eventReports | V exercoreacer 1 | 1.55 | | | |
| | | - eventseports | ▲ eventReport | | | | |
| | | | - eventkepoft | () eventFi Tuples | | | |
| | | | | | | | |
| | | | | V eventsu | _TupleEntry_ | | |
| | | | | ✓ eventField | 4 705 | | |



Tx Model:

| 🖣 data | aTypeDefs | | | | | | | | | |
|--------|-----------------|---|----------------|------------|--------------------|--|------------|--|--|--|
| - | dataTypeDef | | | | | | | | | |
| | () name | Ports | Ports | | | | | | | |
| | () synopsis | Values that can be applied to Incoming and Outoging Port for Condition Forwarding | | | | | | | | |
| | 🔺 atomic | | 1000 C | | CENERAL CONTRACTOR | 3330 | | | | |
| | 104/582/89.04×c | () baseType u8 | | | | | | | | |
| | | A rangeRestriction | | | | | | | | |
| | | | A allowedRange | 8 | | | | | | |
| | | | | = max 0 | | | | | | |
| | - <u></u> | <u></u> | | = min | 9 | | | | | |
| LFB | ClassDefs | | | | | | | | | |
| - | LFBClassDef | | | | | | | | | |
| | = LFBClassID | 4 | | | | | | | | |
| | () name | TxLFB | | | | | | | | |
| | () synopsis | The port Tx LFB for the FE - TVS | | | | | | | | |
| | () version | 1.0 | | | | | | | | |
| | component | s | | | | | | | | |
| | | componer | it (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

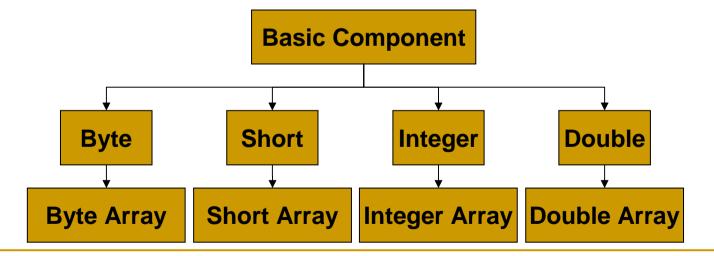
- Java (ver6)
 - Pro's:
 - Easier to code and handle.
 - No use of pointers.
 - Con's
 - Different variable types. (e.g. There are no unsigned types!)
 - No system calls for use with hardware.
- C++ Code (ver11)
 - Pro's
 - System calls.
 - Con's
 - Pointer usage may cause problems.

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:

| Component | Component_Byte | Component_Byte_Array | | |
|--|--|--|--|--|
| Attributes protected int m_iComponentID | Attributes private char m_cByte | <i>Attributes</i> private HashMap <integer,component_byte> Bytes</integer,component_byte> | | |
| protected bool m_bR_W | Operations public Component_Byte(int CompID, bool RW, char data) | Operations public Component_Byte_Array(int CompID, bool RW) | | |
| Operations public Component(int CompID, bool RW) public int get_iComponentID() public char* getValue(char* data) public char* setValue(char* data) public char* getSingleValue(ILV* ilv) public char* delValue(char* data) public char* delSingleValue(ILV* ilv) public char* delSingleValue(ILV* ilv) protected char* getHardwareValue() protected bool setHardwareValue() | Operations redefined from 'Component' public int get_iComponentID() public char* getValue(char* data) public char* setValue(char* data) public char* getSingleValue(ILV* ilv) public char* setSingleValue(ILV* ilv) public char* delValue(char* data) public char* delValue(char* data) public char* delSingleValue(ILV* ilv) protected bool setHardwareValue() protected bool delHardwareValue() protected char* getHardwareValue() | Operations redefined from 'Component_Byte' public int get_iComponentID() public char* getValue(char* data) public char* setValue(char* data) public char* getSingleValue(ILV* ilv) public char* setSingleValue(ILV* ilv) public char* delValue(char* data) public char* delValue(char* data) public char* delSingleValue(ILV* ilv) protected bool setHardwareValue() protected bool delHardwareValue() protected char* getHardwareValue() | | |

Developing ForCES (5)

Complex LFB's:

Component

Attributes protected int m_iComponentID protected bool m_bR_W

 Operations

 public Component(int CompID, bool RW)

 public int get_iComponentID()

 public char* getValue(char* data)

 public char* setValue(char* data)

 public char* getSingleValue(ILV* ilv)

 public char* delValue(char* data)

 public char* delSingleValue(ILV* ilv)

 protected bool setHardwareValue()

 protected bool delHardwareValue()

 protected char* getHardwareValue()

THE THREE THE THREE THREE

Component_Array

Attributes protected Map<int,Component> Components

Operations public Component_Array(int CompID, bool RW) public void Add_Component(Component* Comp)

Operations redefined from 'Component' public int get_iComponentID() public char* getValue(char* data) public char* setValue(char* data) public char* getSingleValue(ILV* ilv) public char* setSingleValue(ILV* ilv) public char* delValue(char* data) public char* delSingleValue(ILV* ilv) protected bool setHardwareValue() protected bool delHardwareValue() protected char* getHardwareValue()

Developing ForCES (6)



ILV Basic_TLV Attributes Attributes private int m ilndex protected short m shType protected short m shLength private int m iLength private char* m pData protected char* m pData protected List<Basic TLV> m INextTLV Operations public ILV() Operations public ILV(char* raw ilv) public Basic TLV() public ILV(int Index, int Length, char* Data) public Basic TLV(char* raw tlv) public Basic_TLV(short Type, short Length, char* Data) public int get ilndex() public int get iLength() public short get shType() public short get shLength() public char* get pData() public char* serialize me() 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()



Developing ForCES (7)



Protocol Message Hierarchy

| | LFBSelect_TLV |
|---|--|
| | Attributes private int m_iLFBClass_ID private int m_iLFBInstance_ID |
| Attributes Attributes protected short m_shType protected short m_shLength protected char* m_pData protected List <basic_tlv> m_INextTLV</basic_tlv> | Operations public LFBSelect_TLV() public LFBSelect_TLV(char* raw_tlv) public LFBSelect_TLV(short Type, short Length, int LFBClassId, int LFBInstanceId, char* Data) public int get_iLFBClass_ID() public int get_iLFBInstance_ID() Operations redefined from 'Basic_TLV' public char* serialize_me() |
| 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() | PathData_TLV Attributes private short m_shFlags private short m_shIDcount private int* m_piIDs |
| public void addTLV() public Basic_TLV* get_NextTLV() public void replace_previousTLV(Basic_TLV* tlv) public int Calculate_TLVs_length() public char* serialize_me() | Operations public PathData_TLV() public PathData_TLV(char* raw_tlv) public PathData_TLV(short Type, short Length, short Flags, short IDcount, int* IDs, char* data) {PathData_TLV} public short get_shFlags() |
| | public short get_shlDcount() public int* get_pilDs() Operations redefined from 'Basic_TLV' public char* serialize_me() |

Developing ForCES (8)

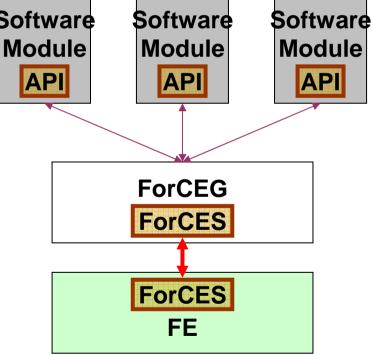


- ForCES is a protocol for configuration of the Forwarding Plane.
- What happens when multiple programs need to configure the same FE? Who controls what?
- Multiple CE controlling an FE or a single manageable CE controlling an FE?
- Need for an Open API between CE and programs for issuing commands.

Developing ForCES (9)

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

Advertise API.

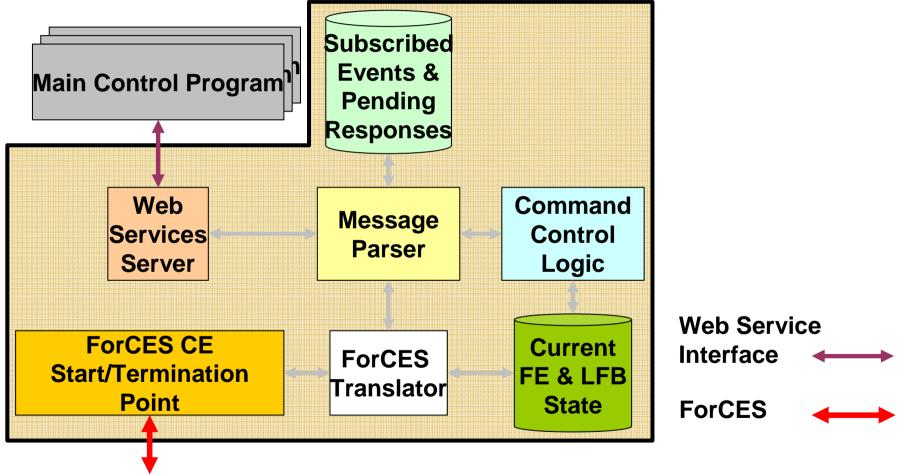




Developing ForCES (10)



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?





Any comments are welcome



Thank you!

