RINASim

Mechanisms and Policies

Vladimír Veselý and Marcel Marek
Brno University of Technology, Czech Republic
veselyv@fit.vutbr.cz
University of Oslo, Norway
marcelma@ifi.uio.no

Agenda

1) Introduction
2) Mechanism and Policy framework
3) Demonstrations
   - Flow Allocator
   - EFCP
4) Conclusion

1) Introduction

Current state

Status Quo

- February 2017 release
  - GitHub repository https://github.com/kvetak/RINA/releases/tag/February2017
  - CDAP API
  - OMNeT++ 5.1 technical candidate compatibility
  - Code refactoring

Virtual Machine

- Out-of-the box virtual machine
  - OMNeT++ 5.0 with the newest RINASim
  - Download from http://nes.fit.vutbr.cz/ivesely/vm/RINASim.zip
  - OVA appliance of MintLinux created on VMWare Workstation
    - *…should work also on VirtualBox and Qemu*
  - Custom highlighter of code syntax

Reporting Issues

- Report on https://github.com/kvetak/RINA/issues

6 Open ✓ 6 Closed Author ▾ Labels ▾ Milestones ▾ Assignee ▾ Sort ▾

- Undeclared signal "Routing-Update" emitted (@signal missing from NED file?). 8
  #19 opened on 7 Nov 2016 by gaixas1
- Left messages with retransmission ON 1
  #17 opened on 21 Oct 2016 by gaixas1
- Compiling RINASim master branch in OMNeT++ 5.0 fails with a lot of c*.h not declared 7
  #16 opened on 9 Oct 2016 by nacarino
- QoS not initialized when searching for existing N-1 flows bug 1
  #14 opened on 22 Jan 2016 by gaixas1
- Src/Dst addresses not initialized on flow requests bug 1
  #13 opened on 21 Jan 2016 by gaixas1
- State vectors are not maintained long enough bug 1
  #12 opened on 15 Jan 2016 by screw

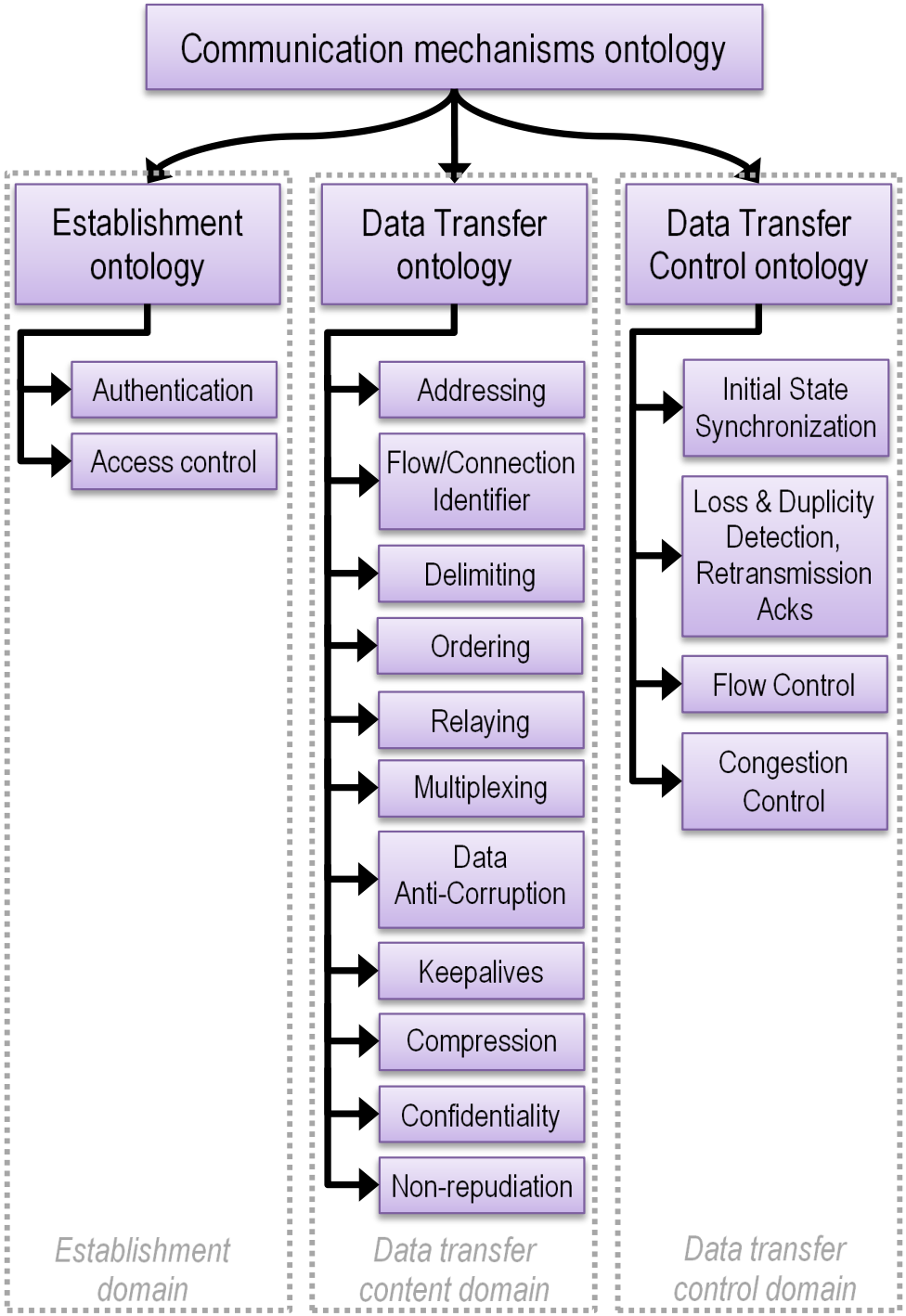
- *Resolution rate will improve…hopefully ☺*

2) Mechanism and Policy

Design notes
A long road from idea/spec towards FSM
Coding conventions

Difference

- **Mechanism**
  - Fixed
  - Cannot be changed
  - E.g., *error checking on data-link layer*
- **Policy**
  - Flexible
  - Can be negotiated
  - E.g., *CRC-32, CRC-64, Vitrebi*



Explicit vs. Implicit Policies

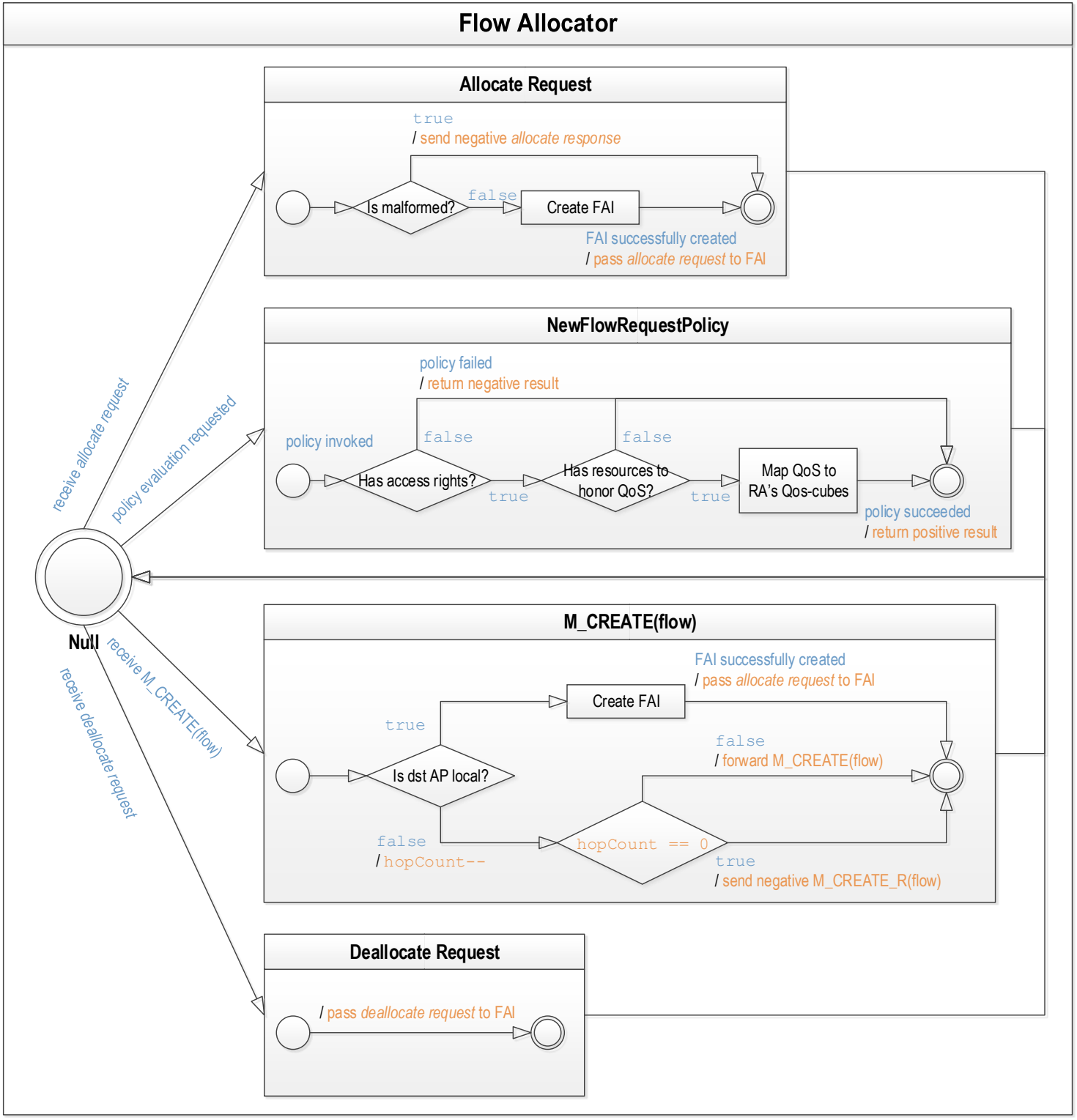
- RINA specs cover explicit policies
  - data processing (e.g., *how is PDU handled by EFCP and RMT*)
  - control processing (e.g., *flow allocation and deallocation procedures*)

- Implicit policies
  - do not have strict placeholder
  - variable inputs and outputs
  - E.g., *routing or secured enrollment*

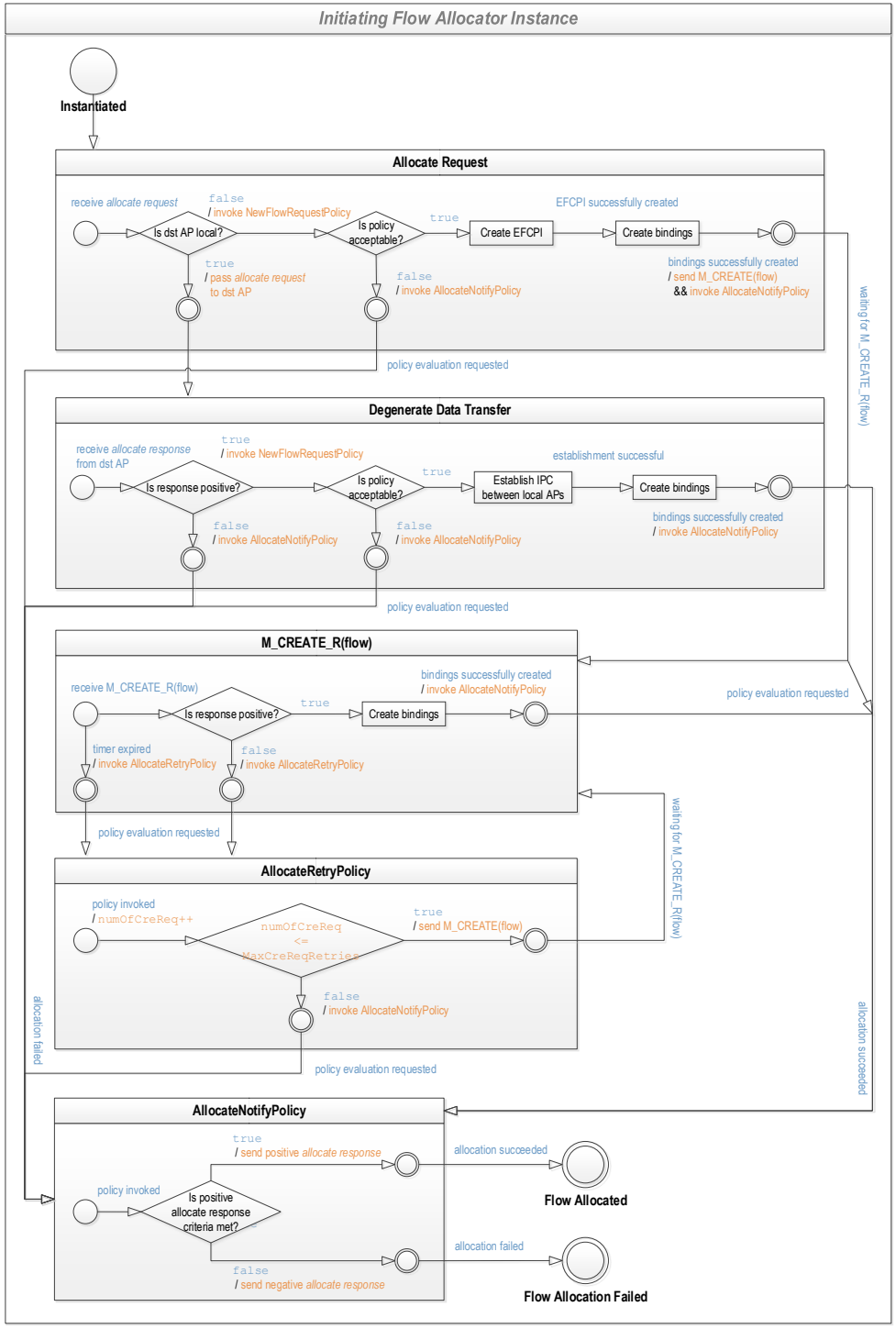
Defining Policy

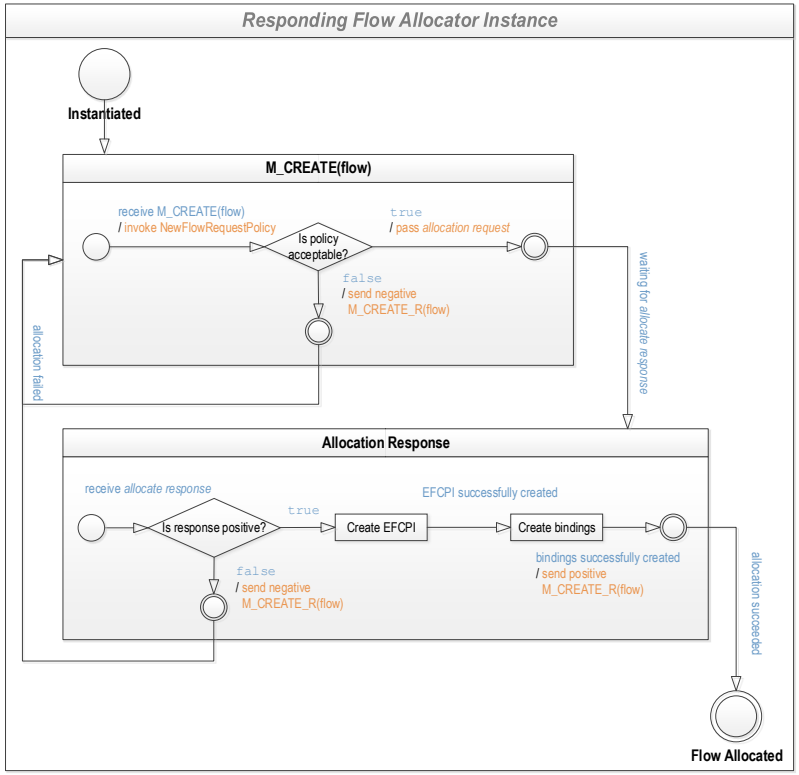
- Policy is (a set of) algorithm(s)
- Properties
  - Deterministic
  - Closed control (*beaware of recursion*)
  - Finite (*avoid state explosion*)
  - Known inputs and outputs
- Description
  - Formal: finite-state machines
  - Implementation: C++ class

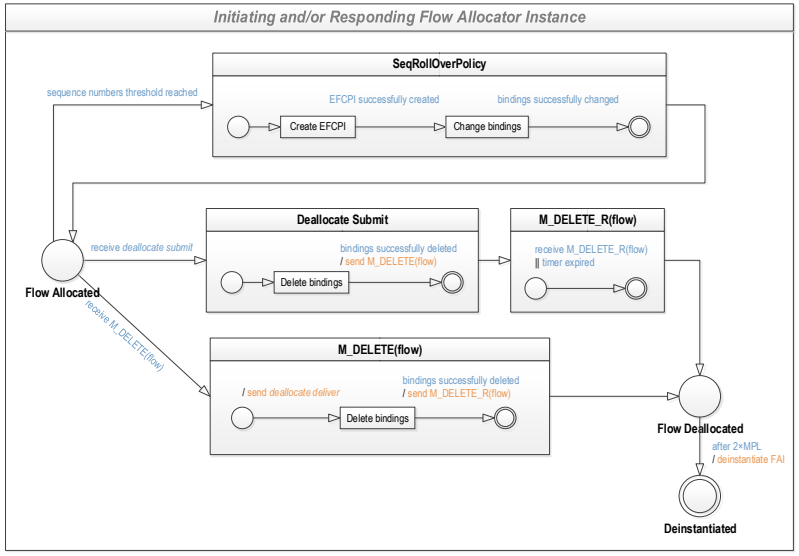
FA


Flow Allocator
Allocate Request
true
/ send negative allocate response
Is malformed?
false
Create FAI
FAI successfully created
/ pass allocate request to FAI
NewFlowRequestPolicy
policy failed
/ return negative result
policy invoked
false
false
Has access rights?
true
Has resources to honor QoS?
true
Map QoS to RA's Qos-cubes
policy succeeded
/ return positive result
receive allocate request
policy evaluation requested
Null
receive M_CREATE(flow)
receive deallocate request
M_CREATE(flow)
FAI successfully created
/ pass allocate request to FAI
Create FAI
true
false
/ forward M_CREATE(flow)
Is dst AP local?
false
/ hopCount--
hopCount == 0
true
/ send negative M_CREATE_R(flow)
Deallocate Request
/ pass deallocate request to FAI

FAI

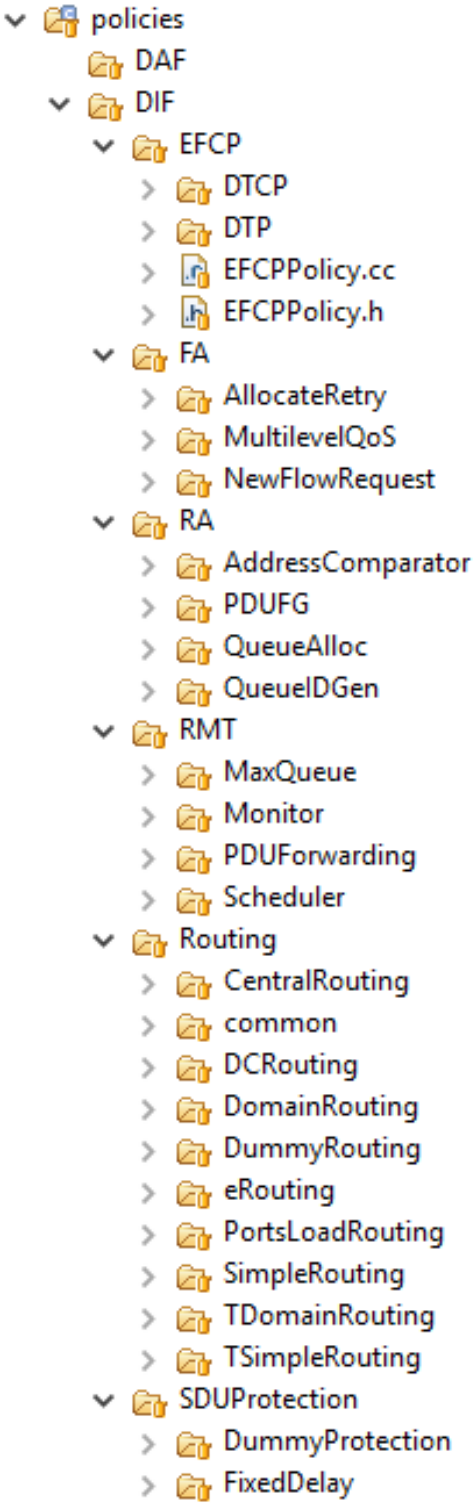

Initiating Flow Allocator Instance
Instantiated
Allocate Request
receive allocate request
Is dst AP local?
false
/ invoke NewFlowRequestPolicy
Is policy acceptable?
true
Create EFCPI
EFCPI successfully created
Create bindings
true
/ pass allocate request to dst AP
false
/ invoke AllocateNotifyPolicy
bindings successfully created
/ send M_CREATE(flow)
&& invoke AllocateNotifyPolicy
policy evaluation requested
waiting for M_CREATE_R(flow)
Degenerate Data Transfer
receive allocate response from dst AP
Is response positive?
true
/ invoke NewFlowRequestPolicy
Is policy acceptable?
true
Establish IPC between local APs
establishment successful
Create bindings
false
/ invoke AllocateNotifyPolicy
false
/ invoke AllocateNotifyPolicy
bindings successfully created
/ invoke AllocateNotifyPolicy
policy evaluation requested
M_CREATE_R(flow)
receive M_CREATE_R(flow)
Is response positive?
true
Create bindings
bindings successfully created
/ invoke AllocateNotifyPolicy
policy evaluation requested
timer expired
/ invoke AllocateRetryPolicy
false
/ invoke AllocateRetryPolicy
policy evaluation requested
waiting for M_CREATE_R(flow)
AllocateRetryPolicy
policy invoked
/ numOfCreReq++
numOfCreReq <= MaxCreReqRetries
true
/ send M_CREATE(flow)
false
/ invoke AllocateNotifyPolicy
allocation failed
policy evaluation requested
allocation succeeded
AllocateNotifyPolicy
policy invoked
Is positive allocate response criteria met?
true
/ send positive allocate response
allocation succeeded
Flow Allocated
false
/ send negative allocate response
allocation failed
Flow Allocation Failed


Responding Flow Allocator Instance
Instantiated
M_CREATE(flow)
receive M_CREATE(flow)
/ invoke NewFlowRequestPolicy
Is policy acceptable?
true
/ pass allocation request
false
/ send negative M_CREATE_R(flow)
allocation failed
waiting for allocate response
Allocation Response
receive allocate response
Is response positive?
true
Create EFCPI
EFCPI successfully created
Create bindings
bindings successfully created
/ send positive M_CREATE_R(flow)
false
/ send negative M_CREATE_R(flow)
allocation succeeded
Flow Allocated


Initiating and/or Responding Flow Allocator Instance
SeqRollOverPolicy
sequence numbers threshold reached
EFCPI successfully created
bindings successfully changed
Create EFCPI
Change bindings
Deallocate Submit
receive deallocate submit
bindings successfully deleted
/ send M_DELETE(flow)
Delete bindings
M_DELETE_R(flow)
receive M_DELETE_R(flow)
|| timer expired
Flow Allocated
receive M_DELETE(flow)
M_DELETE(flow)
/ send deallocate deliver
bindings successfully deleted
/ send M_DELETE_R(flow)
Delete bindings
Flow Deallocated
after 2×MPL
/ deinstantiate FAI
Deinstantiated

RINASim Parts

- Core
  - folder `/src`
  - static `librinasimcore.a` library linked to…
- Policies
  - folder `/policies`
  - dynamic `librinasim.dll` library



Usual Design

1) NED module interface
2) Base class
   - optionally with implicit policy action
3) Policy implementation
   - inheriting all necessary things from base
4) Policy binding
   - with scenario setup in `omnetpp.ini` file

1) NED Module Interface

package rina.policies.DIF.FA.AllocateRetry

IAllocateRetry (interface)

```
23 package rina.policies.DIF.FA.AllocateRetry;
24
25 moduleinterface IAllocateRetry
26 {
27     parameters:
28         @display("i=block/socket");
29 }
```

2) Base class

- Optionally may contain default policy action
  - Not an C++ abstract class in that case
- Every policy has `bool run()` method

omnetpp.ini | AllocateRetryBase.h

```
// The MIT License (MIT)...

#ifndef ALLOCATERETRYBASE_H_
#define ALLOCATERETRYBASE_H_

#include <omnetpp.h>
#include "Common/Utils.h"
#include "Common/Flow.h"

class AllocateRetryBase : public cSimpleModule {
  public:
    virtual bool run(Flow& flow) = 0;

  protected:
    virtual void handleMessage(cMessage *msg) = 0;
    virtual void initialize() = 0;
};

#endif /* ALLOCATERETRYBASE_H_ */
```

3) Policy Implementation

NED file

```
import rina.policies.DIF.FA.AllocateRetry.IAllocateRetry;
package rina.policies.DIF.FA.AllocateRetry.NoRetry;

simple NoRetry like IAllocateRetry
{
    parameters:
        @display("i=block/socket");
}
```

header file

NoRetry.h

rina/policies/DIF/FA/AllocateRetry/NoRetry/NoRetry.h

```
// This program is free software: you can redistribute it and/or modify

#ifndef __RINA_NORETRY_H_
#define __RINA_NORETRY_H_

#include <omnetpp.h>
#include "DIF/FA/AllocateRetry/AllocateRetryBase.h"

using namespace omnetpp;

class NoRetry : public AllocateRetryBase
{
  public:
    virtual bool run(Flow& flow);
  protected:
    virtual void initialize() {setPolicyDisplayString(this);};
    virtual void handleMessage(cMessage *msg) {return;};
};

#endif
```

C++ file

NoRetry.h | NoRetry.cc

```
// This program is free software: you can redistribute it and/or modify

#include "DIF/FA/AllocateRetry/NoRetry/NoRetry.h"

Define_Module(NoRetry);

bool NoRetry::run(Flow& flow) {
    Enter_Method("invokeAllocateRetryPolicy()");
    //Returning false stops any further allocation requests
    return false;
}
```

4) Policy Binding

- Default value somewhere in parent NED file

FlowAllocator.n | RelayAndMux.ned | omnetpp.ini | RMTPort.ned | RMTPortWrapp

rina/src/DIF/FA/FlowAllocator.ned

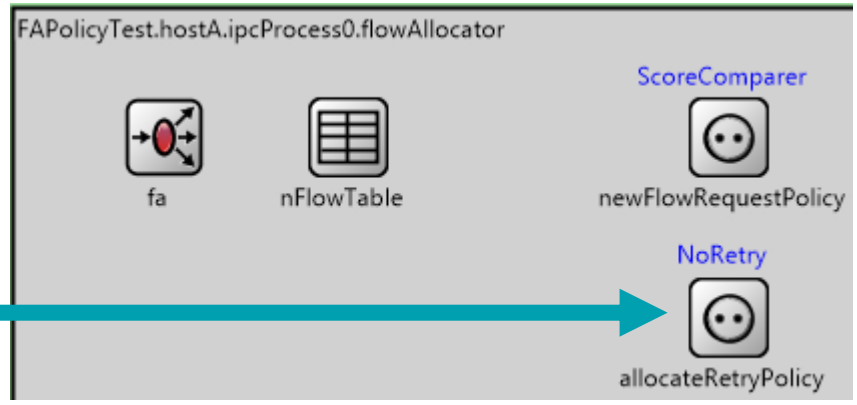
```
28
29
30 module FlowAllocator
31 {
32     parameters:
33         @display("i=block/fork;bgb=424,357");
34         string newFlowReqPolicyType = default("ScoreComparer");
35         string allocRetryPolicyType = default("LimitedRetries");
36         string qosComparerPolicyType = default("QoSIdComparer");
37     gates:
38
39     submodules:
40         allocateRetryPolicy: <allocRetryPolicyType> like IAllocateRetry {
41             @display("p=357,154");
42         }
```

- Override in `omnetpp.ini`

*omnetpp.ini

rina/examples/Webinars/FAPolicyTest/omnetpp.ini

```
5
6 #Enforcing AllocateRetry policy
7 **.flowAllocator.allocRetryPolicyType = "NoRetry"
```



3) Demonstrations

FA
EFCP

Flow Object

Protected Attributes

APNamingInfo	**srcApni** Attribute holding source **APNI**. More...
APNamingInfo	**dstApni** Attribute holding destination **APNI**. More...
int	**srcPortId** Attribute holding source PortId. More...
int	**dstPortId** Attribute holding destination PortId. More...
Address	**srcAddr** Attribute holding source address (initiator of communication) More...
Address	**dstAddr** Attribute holding destination address (end-host for communication) More...
Address	**srcNeighbor** Attribute holding hop-by-hop source address. More...
Address	**dstNeighbor** Attribute holding hop-by-hop destination address. More...
ConnectionId	**conId** Attribute holding **ConnectionId** containing source and destination CEP-Ids and chosen **RA**'s **QoSCube**. More...
uint32_t	**createFlowRetries** Attribute holding current number of allowed CreateFlow retries. More...
uint32_t	**maxCreateFlowRetries** Attribute holding maximum allowed number of CreateFlow retries. More...
uint32_t	**hopCount** Attribute holding flow's hop-count. More...
QoSCube	**qosCube** Attribute holding the assigned **QoSCube**. More...
QoSReq	**qosReqs** Attribute holding wanted QoS parameters in form of **QoSReq**. More...
long	**allocInvokeId** Attribute holding persistant InvokeId used for allocation. More...
long	**deallocInvokeId** Attribute holding persistant InvokeId used for deallocation. More...
bool	**ddtFlag** Attribute representing whether flow is for Degenerate **Data** Transfer or not. More...

Public Member Functions

	Flow () Constructor for the flow with undefined values. More...
	Flow (**APNamingInfo** src, **APNamingInfo** dst) Default constructor for flow between two applications. More...
virtual	**~Flow** () Destructor assigning default uninitialized values. More...
bool	**operator==** (const **Flow** &other) const Equal operator overloading. More...
bool	**compare** (const **Flow** &other) const Comparator taking into account src/dst addresses, src/dst APNIs, src-dst ConIds/PortIds. More...
virtual **Flow** *	**dup** () const **Flow** object duplication method that creates copy with. More...
Flow *	**dupToMgmt** () const
std::string	**info** () const Prints **Flow** information as string Calls variety of other info functions to produce final output. More...
std::string	**infoSource** () const Prints **Flow** source information as string Outputs source **APNI**, address, neigbor-address, PortId, CEP-Id. More...
std::string	**infoDestination** () const Prints **Flow** destination information as string Outputs destination **APNI**, address, neigbor-address, PortId, CEP-Id. More...
std::string	**infoOther** () const Prints **Flow** create flow information and hop-cpunt as string. More...
std::string	**infoQoS** () const Prints **RA**'s QoSCube-id that **FA** choosed during **Flow** allocation phase Accompanied QoSParameters could be find in **A**

Protected Member Functions

void	**swapPortIds** () Auxiliary function swapping source and destination PortIds. More...
void	**swapAddresses** () Auxiliary function swapping source and destination Addresses. More...
void	**swapCepIds** () Auxiliary function swapping source and destination CEP-Ids. More...
void	**swapApni** () Auxiliary function swapping source and destinatio **APNI**. More...

FA: Demo

- /examples/Webinars/FAPolicyTest

```
import rina.policies.DIF.FA.AllocateRetry.IAllocateRetry;
package rina.policies.DIF.FA.AllocateRetry.UnlimitedRetries;

simple UnlimitedRetries like IAllocateRetry
{
    parameters:
        @display("i=block/socket");
}
```

UnlimitedRetries.cc

```
// This program is free software: you can redistribute it and/or modify

#include "DIF/FA/AllocateRetry/UnlimitedRetries/UnlimitedRetries.h"

Define_Module(UnlimitedRetries);

bool UnlimitedRetries::run(Flow& flow) {
    Enter_Method("invokeAllocateRetryPolicy()");
    //Increment number of retries
    flow.setCreateFlowRetries( flow.getCreateFlowRetries() + 1 );
    //Allow unlimited number of retries
    return true;
}
```

omnetpp.ini | omnetpp.ini | UnlimitedRetrie | EFCPPolicyTes

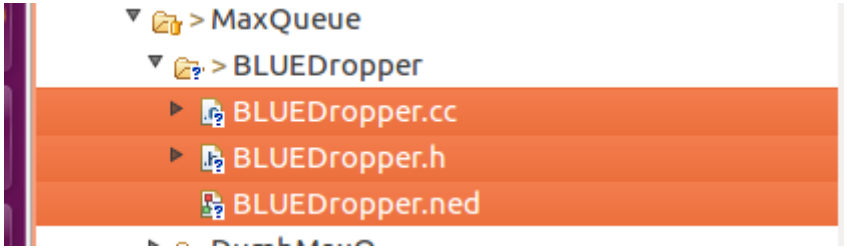
```
// This program is free software: you can redistribute it and/

#ifndef __RINA_UNLIMITEDRETRIES_H_
#define __RINA_UNLIMITEDRETRIES_H_

#include <omnetpp.h>
#include "DIF/FA/AllocateRetry/AllocateRetryBase.h"

using namespace omnetpp;

class UnlimitedRetries : public AllocateRetryBase
{
  public:
    virtual bool run(Flow& flow);
  protected:
    virtual void initialize() {setPolicyDisplayString(this);};
    virtual void handleMessage(cMessage *msg) {return;};
};

#endif
```

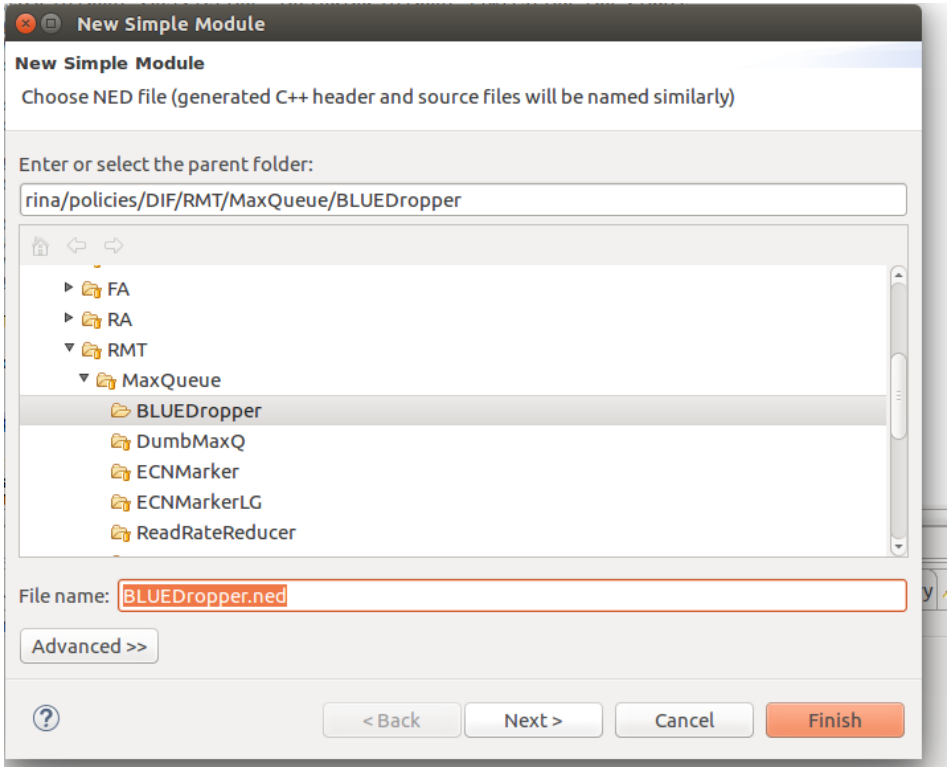
```
#Enforcing AllocateRetry policy
[Config Ping-Default]
**.flowAllocator.allocRetryPolicyType = "LimitedRetries"

[Config Ping-NoRetry]
**.flowAllocator.allocRetryPolicyType = "NoRetry"

[Config Ping-UnlimitedRetries]
**.flowAllocator.allocRetryPolicyType = "UnlimitedRetries"
```

EFCP: Congestion Demo ①

- /examples/Webinars/EFCPPolicyTest

```
    }
connections:
    hostA.medium <--> DatarateChannel {  datarate = 100Mbps; delay = 100us; ber = 0; } <--> switch.medium[0];
    switch.medium[1] <--> DatarateChannel {  datarate = 100Mbps; delay = 5s; ber = 0; } <--> hostB.medium;
```





EFCP: Congestion Demo ②

```
//
import rina.policies.DIF.RMT.MaxQueue.IntRMTMaxQPolicy;

package rina.policies.DIF.RMT.MaxQueue.BLUEDropper;


//
// TODO auto-generated module
//
simple BLUEDropper like IntRMTMaxQPolicy
{

        parameters:
        @display("i=block/socket");
        @signal[RMT-SlowDownRequest];

        double dropProbability = default(0.4);
        bool marking = default(false);
}
```

```
#include <omnetpp.h>

#include "RMTMaxQBase.h"
#include "REDMonitor.h"

using namespace omnetpp;

class BLUEDropper : public RMTMaxQBase
{
public:
  virtual bool run(RMTQueue* queue);

  private:
    void onPolicyInit();
    bool dropOrMark(RMTQueue* queue);
    REDMonitor* monitor;
};
```

```
[Config CongestionPing]
fingerprint = "0000-0000"


**.hostA.applicationProcess1.AEMonitor.**.iae.size = 256B


**.hostA.applicationProcess1.apInst.dstApName = "DestinationB"
**.hostA.applicationProcess1.apInst.startAt = 10s
**.hostA.applicationProcess1.apInst.interval = 0.5s
**.hostA.applicationProcess1.apInst.stopAt = 250s


#Congestion parameters
**.switch.relayIpc.relayAndMux.defaultMaxQLength = 5
**.switch.relayIpc.relayAndMux.defaultThreshQLength = 3
**.switch.relayIpc.relayAndMux.maxQPolicyName = "BLUEDropper"
**.switch.relayIpc.relayAndMux.qMonitorPolicyName = "REDMonitor"

**.efcp.rtt = 25s
**.host1.ipcProcess1.efcp.efcp.initialSenderCredit = 50
**.switch.ipcProcess1.efcp.efcp.initialSenderCredit = 3
**.host2.ipcProcess0.efcp.efcp.rcvCredit = 3
**.switch.ipcProcess1.efcp.efcp.maxClosedWinQueLen = 4
```

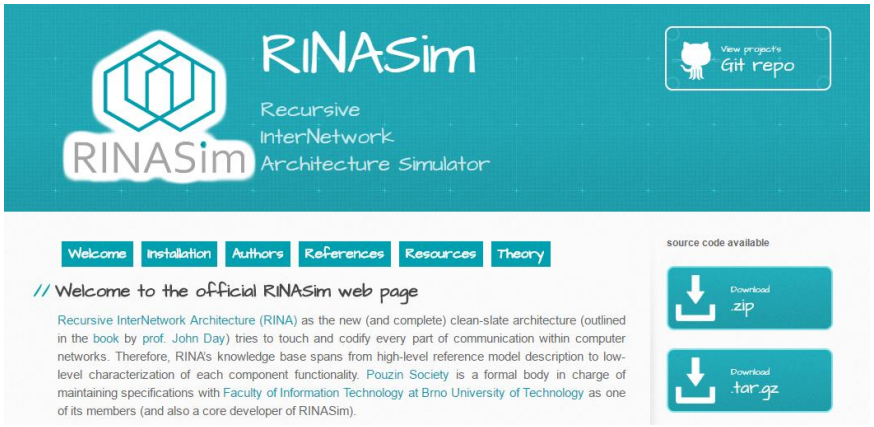
```
            <CostTime>0</CostTime>
            <CostBits>0</CostBits>
            <ATime>4</ATime>
            <RxOn>1</RxOn>
            <WinOn>1</WinOn>
            <RateOn>0</RateOn>
        </QoSCube>
```

4) Conclusion

Final remarks
Open call

Need Help?

- Check the official webpage
  - Visit https://rinasim.omnetpp.org



- Skype group chat
  - skype:?chat&blob=-bdq6qH_uDXIlbRk_4_XwqZyplfXPl4IzCq4P-S0BrsttjgPR8CNJKV9-Yyn1TYopaYZD2g3bIC_Yv0C
  - https://join.skype.com/B9Tt5aTPd0nC

- Sign to mailing-list rinasim@fit.vutbr.cz
  - Use http://www.fit.vutbr.cz/mailman/listinfo/rinasim

Team

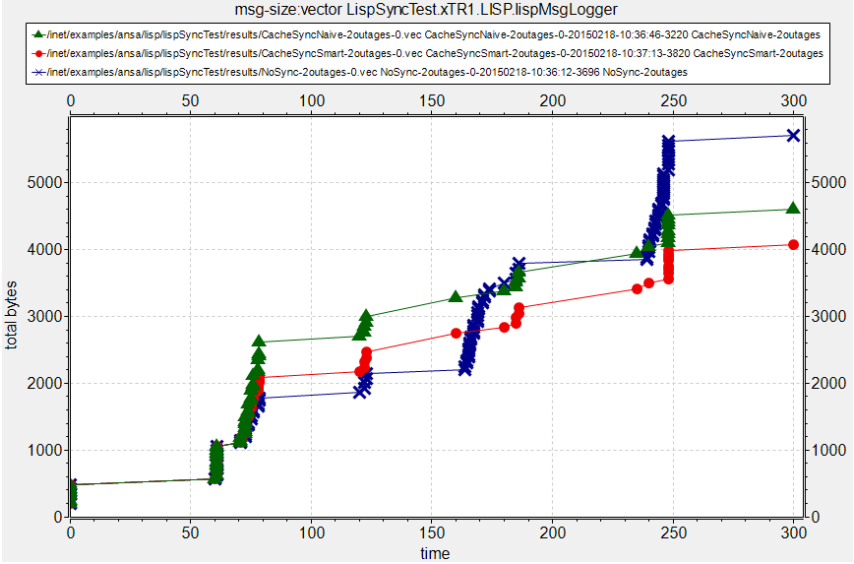
- RINASim is a joint work of following people
  - Vladimír Veselý (@kvetak)
  - Marcel Marek (@screw)
  - Kamil Jeřábek (@kjerabek)
  - Tomáš Hykel (@thykel)
  - Sergio Leon Gaixas (@gaixas1)
  - Peyman Teymoori (@peyman-t)
  - Ehsan Elahi (@ehsanzahoor)
  - Kewin Rausch (@kewinrausch)
  - Fatma Hrizi (@fatmahrizi)
  - Kleber Leal (@kaleal)
- Green marked individual are usually willing to deal with your RINASim troubles

Problems for Today and Tomorrow

1) Add more source code comments
2) Improve Doxygen documentation
3) Create real data-link layer simulation modules
4) Extend RIB functionality

Topics for Other Webinars

- *What did we not covered?*
  - Results gathering and analysis



- *Are you interested about work of others?*
- *Suggest your topic…*

RINASim

Thank you!
Good morning / Good night ☺