

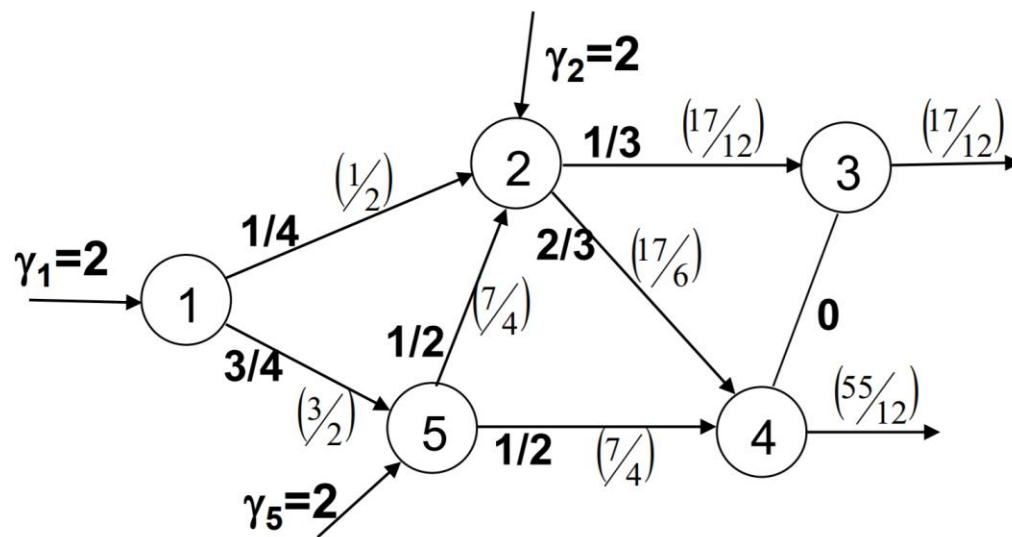
OMNet++ - Caso 3

Rendimiento en Redes de Telecomunicación

Adrián Santiago Santo-Tomás

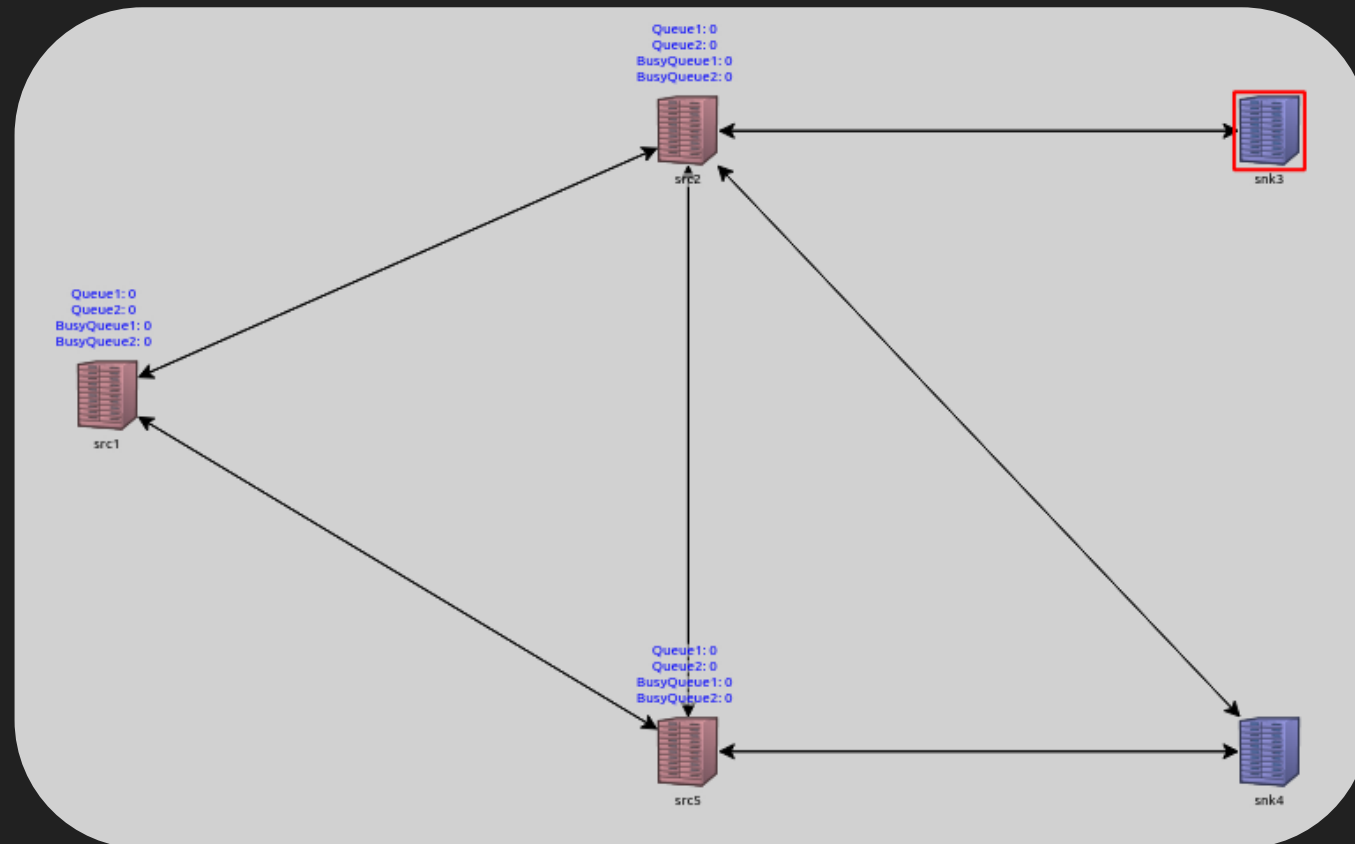
2020-2021

Red a desarrollar



$$\mu_i = 3 \quad \forall i$$

Red desarrollada



Fases de desarrollo

Fase 0:

- Conceptualización
- Definición objetivos
- Planificación
- Entorno de desarrollo OMNet++
- Documentación y API OMNet++



Fase 1:

- Desarrollo básico de módulos
- Definición de la red
- Conexión de componentes y simulación básica



Fase 2:

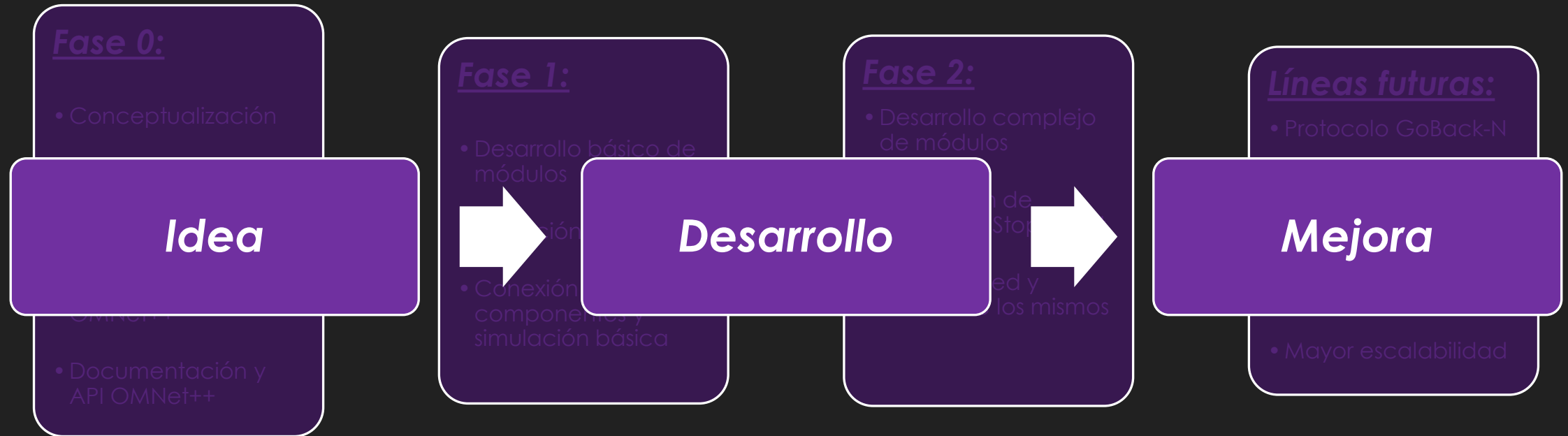
- Desarrollo complejo de módulos
- Simulación de protocolo Stop&Wait
- Datos de red y análisis de los mismos



Líneas futuras:













- Protocolo GoBack-N
- Exportación y análisis de datos de red
- Depuración de código
- Mayor escalabilidad

Fases de desarrollo



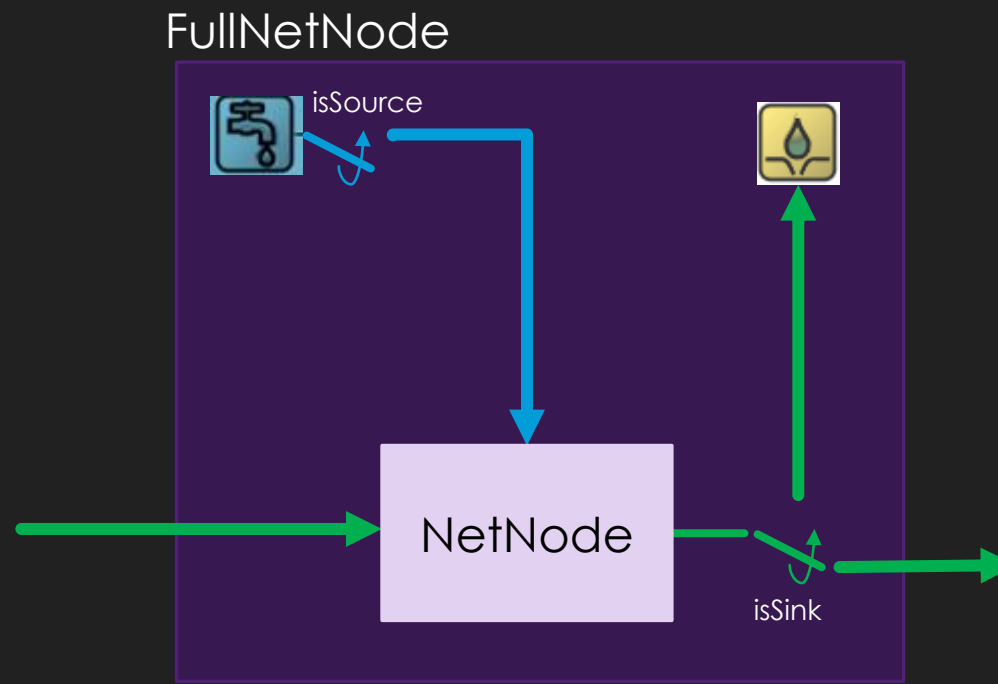
Características fundamentales:

- Modularidad

	FullNetNode.cc
	FullNetNode.h
	FullNetNode.ned
	Makefile
	NetNode.cc
	NetNode.h
	NetNode.ned
	NetworkDefines.h
	asstPacket.msg
	asstPacket_m.cc
	asstPacket_m.h
	package.ned

Características fundamentales:

- Modularidad



Características fundamentales:

- Modularidad
- Parametrización

*.ini

```
[General]
network = RTT_CAS03_NET
cpu-time-limit = 60s
#debug-on-errors = true

**.src*.isSource = true
**.snk*.isSink = true

#not protocol
***.src*.protocol = 0
***.snk*.protocol = 0
#stop&wait
**.src*.protocol = 1
**.snk*.protocol = 1

**.src*.mu_mean = 3
**.snk*.mu_mean = 3

**.src*.lambda_mean = 2

**.src1.probRutado = 0.25
**.src2.probRutado = 0.33
**.src5.probRutado = 0.25
**.snk3.probRutado = 1
**.snk4.probRutado = 1
```

*.h

```
/*
 * NetworkDefines.h
 *
 * Created on: Jan 1, 2021
 * Author: adri
 */

#ifndef NETWORKDEFINES_H_
#define NETWORKDEFINES_H_

#define PROTOCOLO_TX_NETNODES 0
#define PROBABILIDAD_ERROR_NETNODES 0.05
#define PROBABILIDAD_ROUTE_NETNODES 0.7

#define ACCION_NADA 0
#define ACCION_ENVIAR 1
#define ACCION_ENVIO_OK 2
#define ACCION_ENVIO_NOK 3
#define ACCION_REENVIAR 4
#define ACCION_ENVIAR_Q1 5
#define ACCION_ENVIO_OK_Q1 6
#define ACCION_ENVIO_NOK_Q1 7
#define ACCION_REENVIAR_Q1 8
#define ACCION_ENVIAR_Q2 9
#define ACCION_ENVIO_OK_Q2 10
#define ACCION_ENVIO_NOK_Q2 11
#define ACCION_REENVIAR_Q2 12

#define PROTOCOLO_TX_NOT_PROTOCOL 0
#define PROTOCOLO_TX_STOP_AND_WAIT 1
#define PROTOCOLO_TX_GO_BACK_N 2

#define MESSAGE_KIND_FROM_SOURCE 1
#define MESSAGE_KIND_ACK 2
#define MESSAGE_KIND_NACK 3
#define MESSAGE_KIND_PACKET 4
#define MESSAGE_KIND_CORRUPTED 5

#define GATE_INDEX_1 0
#define GATE_INDEX_2 1

#define TIEMPO_MEDIO_ENTRE_SERVICIOS 0.333
#define TIEMPO_MEDIO_ENTRE_LLEGADAS 0.5
#define TAM_MEDIO_PAQUETES_BITS 1000

#define PACKET_TIME_TO_LIVE 10

#define DESCRIPCION_MSG_SERVICETIME "serviceTime"
#define DESCRIPCION_MSG_ARRIVALTIME "arrivalTime"
#define DESCRIPCION_MSG_PACKET "packet"

#define OUTPUT_GATE_FREE 0
#define OUTPUT_GATE_BUSY 1

#endif /* NETWORKDEFINES_H_ */
```


*.CC

Características fundamentales:

- Modularidad
- Parametrización
- Flexibilidad:
 - Desarrollo
 - Simulación

```
void FullNetNode::initialize()
{
    //Inicializacion de nodo por defecto
    NetNode::initialize();

    //Inicializacion de tasas de llegadas y servicios
    if (par("lambda_mean").doubleValue())
    {
        setTiempoMedioEntreLlegadas(par("lambda_mean").doubleValue());
    }
    else
    {
        setTiempoMedioEntreLlegadas(TIEMPO_MEDIO_ENTRE_LLEGADAS);
    }
    if (par("mu_mean").doubleValue())
    {
        setTiempoMedioEntreServicios(par("mu_mean").doubleValue());
    }
    else
    {
        setTiempoMedioEntreServicios(TIEMPO_MEDIO_ENTRE_SERVICIOS);
    }

    //Inicializacion de tipo de nodo: Fuente, Sumidero, Ambas o Ninguna
    if (par("isSource").boolValue())
    {
        AsstPacket *msg_arrival = new AsstPacket(DESCRIPCION_MSG_ARRIVALTIME);
        scheduleAt(simTime() + exponential(getTiempoMedioEntreLlegadas()), msg_arrival);
        setisSource(1);
    }
    if (par("isSink").boolValue())
    {
        setisSink(1);
    }

    //Inicializacion de parametros de nodo
    if (par("protocol").intValue())
    {
        setprotocolType(par("protocol").intValue());
    }
    if (par("probRutado").doubleValue())
    {
        setpRoute(par("probRutado").doubleValue());
    }

    //Numero de paquetes a generar
    if (par("numPackets").intValue())
    {
        setNumPacketsToGenerate(par("numPackets").intValue());
    }
}
```

Características fundamentales:

- Modularidad
- Parametrización
- Flexibilidad:
 - Desarrollo
 - Simulación

*.ned

```
simple FullNetNode
{
    parameters:
        bool    isSource = default(false);    //indica si es fuente de paquetes (origen)
        double  lambda_mean = default(1);    //si es fuente, media de tasa de llegadas
        bool    isSink = default(false);    //indica si es sumidero de paquetes (destino)
        double  mu_mean = default(1);    //tasa media de servicios
        int     protocol = default(0);    //Tipo protocolo utilizado: none, S&W o GBN
        double  probRutado = default(0.5);    //Probabilidad de envio por output 1 del nodo
        int     numPackets = default(10);    //Numero de paquetes maximo a generar en caso de fuente

    gates:
        input  inBack[];
        input  inForward[];
        output outBack[];
        output outForward[];
}
```

[General]

```
network = RTT_CASO3_NET
cpu-time-limit = 60s
#debug-on-errors = true
```

```
**src*.isSource = true
**snk*.isSink = true
```

```
#not protocol
**src*.protocol = 0
**snk*.protocol = 0
#stop&wait
**src*.protocol = 1
**snk*.protocol = 1
```

```
**src*.mu_mean = 3
**snk*.mu_mean = 3
```

```
**src*.lambda_mean = 2
```

```
**src1.probRutado = 0.25
**src2.probRutado = 0.33
**src5.probRutado = 0.25
**snk3.probRutado = 1
**snk4.probRutado = 1
```

*.ini

Características fundamentales:

- Modularidad
- Parametrización
- Flexibilidad:
 - Desarrollo
 - Simulación
- Amplia funcionalidad para implementación de red

*.ned

```
network RTT_CASO3_NET
{
    types:
        channel C extends ned.DatarateChannel {
            datarate = 1Mbps;
            delay = 100us;
            //per = 0.1;
        }
    submodules:
        src1: FullNetNode {
            //@display("i=block/source,red");
            @display("i=device/router,red;p=50,130");
            @display("t=Queue1: 0\nQueue2: 0\nBusyQueue1: 0\nBusyQueue2: 0;tt= Node1(Source)");
        }
        src2: FullNetNode {
            @display("i=device/router,red;p=200,62");
            @display("t=Queue1: 0\nQueue2: 0\nBusyQueue1: 0\nBusyQueue2: 0;tt=Node2(Source)");
        }
        src5: FullNetNode {
            @display("i=device/router,red;p=200,222");
            @display("t=Queue1: 0\nQueue2: 0\nBusyQueue1: 0\nBusyQueue2: 0;tt=Node5(Source)");
        }
        snk3: FullNetNode {
            @display("i=device/router,blue;p=350,62");
            @display("tt=Node3(Sink)");
            //@display("t=Queue1: 0\nQueue2: 0;tt=Node3(Sink)");
        }
        snk4: FullNetNode {
            @display("i=device/router,blue;p=350,222");
            @display("tt=Node4(Sink)");
            //@display("t=Queue1: 0\nQueue2: 0;tt=Node4(Sink)");
        }
    }
    connections:

        //Real channels
        //CX: 1-2
        src1.outFordward++ --> C --> src2.inBack++;
        src1.inFordward++ <-- C <-- src2.outBack++;

        //CX: 1-5
        src1.outFordward++ --> C --> src5.inBack++;
        src1.inFordward++ <-- C <-- src5.outBack++;

        //CX: 2-3
        src2.outFordward++ --> C --> snk3.inBack++;
        src2.inFordward++ <-- C <-- snk3.outBack++;

        //CX: 2-4
        src2.outFordward++ --> C --> snk4.inBack++;
        src2.inFordward++ <-- C <-- snk4.outBack++;

        //CX: 5-2
        src5.outFordward++ --> C --> src2.inBack++;
        src5.inFordward++ <-- C <-- src2.outBack++;

        //CX: 5-4
        src5.outFordward++ --> C --> snk4.inBack++;
        src5.inFordward++ <-- C <-- snk4.outBack++;
}
```

cPacket (personalizado)

- Analizar path
- Analizar tiempos
- Controlar tamaño mediante relleno:
 - Distribución exponencial

*.msg

```
1  //
2  // Created on: Jan 27, 2021
3  // Author: adri
4  //
5
6  namespace asst_rtt_caso3;
7
8  //
9  // TODO generated message class
10 //
11 packet AsstPacket {
12     int        numHops;
13     string      pktName;
14     double      sourceTime;
15     double      sinkTime;
16     string      pktPath;
17     char        relleno[]; // 1byte por char
18 }
```

Funciones fundamentales

- Parametrización
- Flexibilidad
- Escalabilidad y adición de funcionalidades

```
rc = processMessage(packet, protocolType, &action);
if(rc != 0)
{
    EV << "ERROR: processMessage() \n";
    return;
}

rc = sendMessage(protocolType, action);
if(rc != 0)
{
    EV << "ERROR: sendMessage() \n";
    return;
}
```

```

int NetNode::processMessage(AsstPacket *msg, int protocolType, int *action)
{
    //Procesamiento de mensajes recibidos
    int rc = 0;
    (*action) = ACCION_NADA;

    switch(protocolType)
    {
        case PROTOCOLO_TX_STOP_AND_WAIT:
            rc = processMessageStopAndWait(msg, action);
            if(rc != 0)
            {
                EV << "ERROR: processMessageStopAndWait() \n";
                return rc;
            }
            break;
        case PROTOCOLO_TX_GO_BACK_N:
            rc = processMessageGoBackN(msg, action);
            if(rc != 0)
            {
                EV << "ERROR: processMessageGoBackN() \n";
                return rc;
            }
            break;
        case PROTOCOLO_TX_NOT_PROTOCOL:
        default:
            rc = processMessageNotProtocol(msg, action);
            if(rc != 0)
            {
                EV << "ERROR: processMessageNotProtocol() \n";
                return rc;
            }
            break;
    }

    char charT[50];
    memset(charT, 0, 50);
    sprintf(charT, "Queue1: %d\nQueue2: %d\nBusyQueue1: %d\nBusyQueue2: %d", (int)messageQueue1.getSize(), (int)messageQueue2.getSize(), (int)busyQueue1.getSize(), (int)busyQueue2.getSize());
    getDisplayString().setTagArg("t", 0, charT);

    return rc;
}

```

```

int NetNode::sendMessage(int protocolType, int action)
{
    //Envío de mensajes
    int rc = 0;

    switch(protocolType)
    {
        case PROTOCOLO_TX_STOP_AND_WAIT:
            rc = sendMessageStopAndWait(action);
            if(rc != 0)
            {
                EV << "ERROR: sendMessageStopAndWait() \n";
                return rc;
            }
            break;
        case PROTOCOLO_TX_GO_BACK_N:
            rc = sendMessageGoBackN(action);
            if(rc != 0)
            {
                EV << "ERROR: sendMessageGoBackN() \n";
                return rc;
            }
            break;
        case PROTOCOLO_TX_NOT_PROTOCOL:
        default:
            rc = sendMessageNotProtocol(action);
            if(rc != 0)
            {
                EV << "ERROR: sendMessageNotProtocol() \n";
                return rc;
            }
            break;
    }

    return rc;
}

```

Definición de protocolos mediante configuración

*.ini

```
[General]
network = RTT_CAS03_NET
cpu-time-limit = 60s
#debug-on-errors = true

**.src*.isSource = true
**.snk*.isSink = true

#not protocol
**.src*.protocol = 0
**.snk*.protocol = 0

#stop&wait
**.src*.protocol = 1
**.snk*.protocol = 1

**.src*.mu_mean = 3
**.snk*.mu_mean = 3

**.src*.lambda_mean = 2

**.src1.probRutado = 0.25
**.src2.probRutado = 0.33
**.src5.probRutado = 0.25
**.snk3.probRutado = 1
**.snk4.probRutado = 1
```

*.h

```
/*
 * NetworkDefines.h
 *
 * Created on: Jan 1, 2021
 * Author: adri
 */

#ifndef NETWORKDEFINES_H_
#define NETWORKDEFINES_H_

#define PROTOCOLO_TX_NETNODES 0
#define PROBABILIDAD_ERROR_NETNODES 0.05
#define PROBABILIDAD_ROUTE_NETNODES 0.7

#define ACCION_NADA 0
#define ACCION_ENVIAR 1
#define ACCION_ENVIO_OK 2
#define ACCION_ENVIO_NOK 3
#define ACCION_REENVIAR 4
#define ACCION_ENVIAR_Q1 5
#define ACCION_ENVIO_OK_Q1 6
#define ACCION_ENVIO_NOK_Q1 7
#define ACCION_REENVIAR_Q1 8
#define ACCION_ENVIAR_Q2 9
#define ACCION_ENVIO_OK_Q2 10
#define ACCION_ENVIO_NOK_Q2 11
#define ACCION_REENVIAR_Q2 12

#define PROTOCOLO_TX_NOT_PROTOCOL 0
#define PROTOCOLO_TX_STOP_AND_WAIT 1
#define PROTOCOLO_TX_GO_BACK_N 2

#define MESSAGE_KIND_FROM_SOURCE 1
#define MESSAGE_KIND_ACK 2
#define MESSAGE_KIND_NACK 3
#define MESSAGE_KIND_PACKET 4
#define MESSAGE_KIND_CORRUPTED 5

#define GATE_INDEX_1 0
#define GATE_INDEX_2 1

#define TIEMPO_MEDIO_ENTRE_SERVICIOS 0.333
#define TIEMPO_MEDIO_ENTRE_LLEGADAS 0.5
#define TAM_MEDIO_PAQUETES_BITS 1000

#define PACKET_TIME_TO_LIVE 10

#define DESCRIPCION_MSG_SERVICETIME "serviceTime"
#define DESCRIPCION_MSG_ARRIVALTIME "arrivalTime"
#define DESCRIPCION_MSG_PACKET "packet"

#define OUTPUT_GATE_FREE 0
#define OUTPUT_GATE_BUSY 1

#endif /* NETWORKDEFINES_H_ */
```

Definición de acciones mediante tipo de paquete (setKind(), getKind())

// Arrivals (generation at source)

```
case MESSAGE_KIND_FROM_SOURCE:  
    ACCION_ENVIAR;
```

// Received OK packet tx. ACK:

```
case MESSAGE_KIND_ACK:  
    if GATE_INDEX_1  
        ACCION_ENVIO_OK_Q1;  
    if GATE_INDEX_2  
        ACCION_ENVIO_OK_Q2;
```

// Received NotOK packet tx. NACK:

```
case MESSAGE_KIND_NACK:  
    if GATE_INDEX_1  
        ACCION_ENVIAR_Q1;  
    if GATE_INDEX_2  
        ACCION_ENVIAR_Q2;
```

// Arrivals (from other nodes)

```
case MESSAGE_KIND_PACKET:  
    if GATE_INDEX_1  
        ACCION_REENVIAR_Q1;  
    if GATE_INDEX_2  
        ACCION_REENVIAR_Q2;
```

// Arrivals (from other nodes) with errors

```
MESSAGE_KIND_CORRUPTED:  
    if GATE_INDEX_1  
        ACCION_ENVIO_NOK_Q1;  
    if GATE_INDEX_2  
        ACCION_ENVIO_NOK_Q2;
```


Comunicación mediante acciones

//Send PACKETS from sources

case ACCION_ENVIAR:

...

case ACCION_ENVIAR_Q1:

...

case ACCION_ENVIAR_Q2:

...

//Send PACKETS (routing)

case ACCION_REENVIAR_Q1:

...

case ACCION_REENVIAR_Q2:

...

//RX. ACK's

case ACCION_ENVIO_OK_Q1:

...

case ACCION_ENVIO_OK_Q2:

...

//TX. NACK's

case ACCION_ENVIO_NOK_Q1:

...

case ACCION_ENVIO_NOK_Q2:

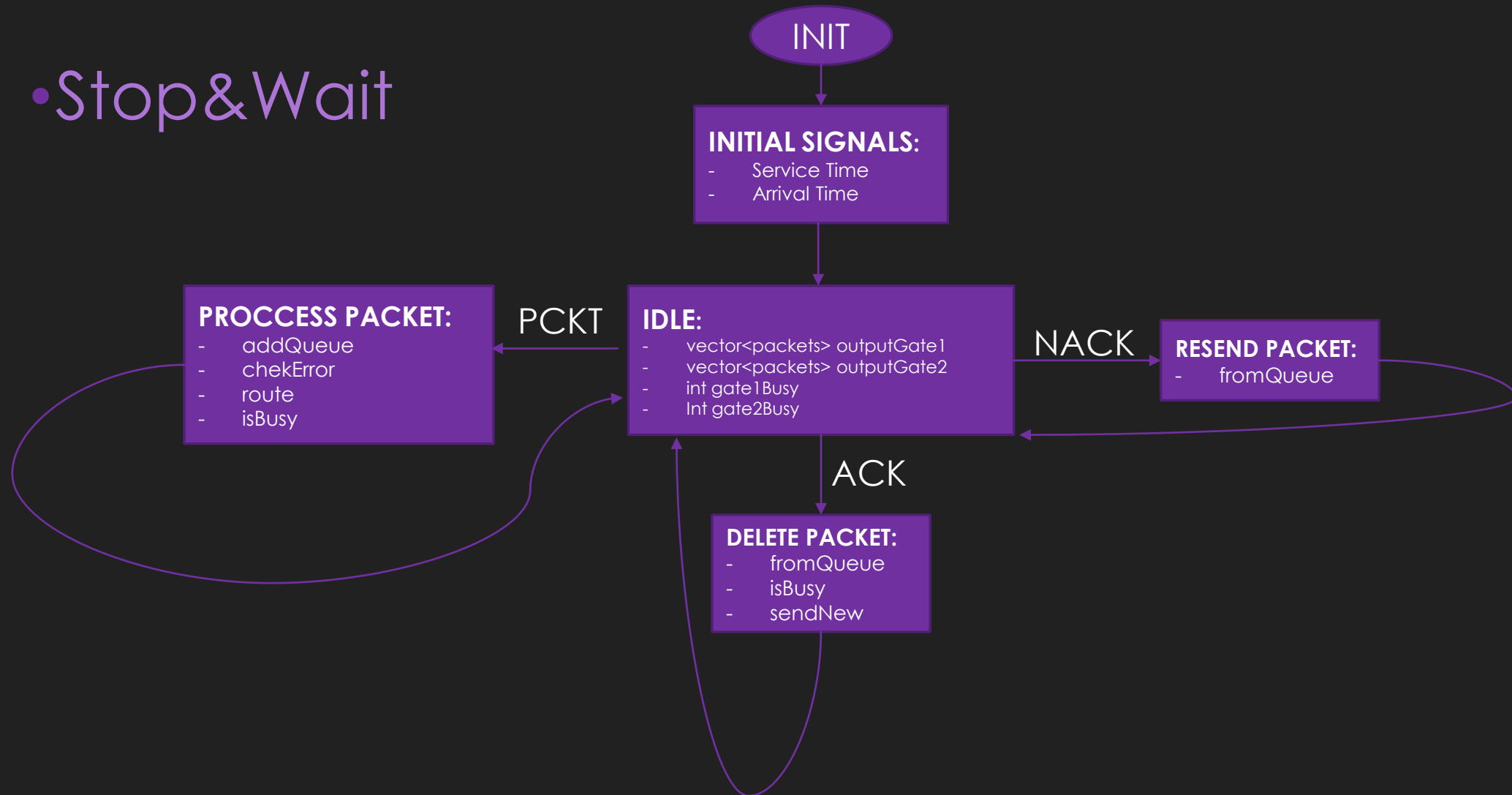
...

Máquina de estados

- Concebida de manera indirecta:
 - Planteamiento inicial:
 - Evento->Acción
- Particular de cada protocolo
- Gestión de llegada de paquetes, tipo de los mismos y generación de respuestas adecuadas

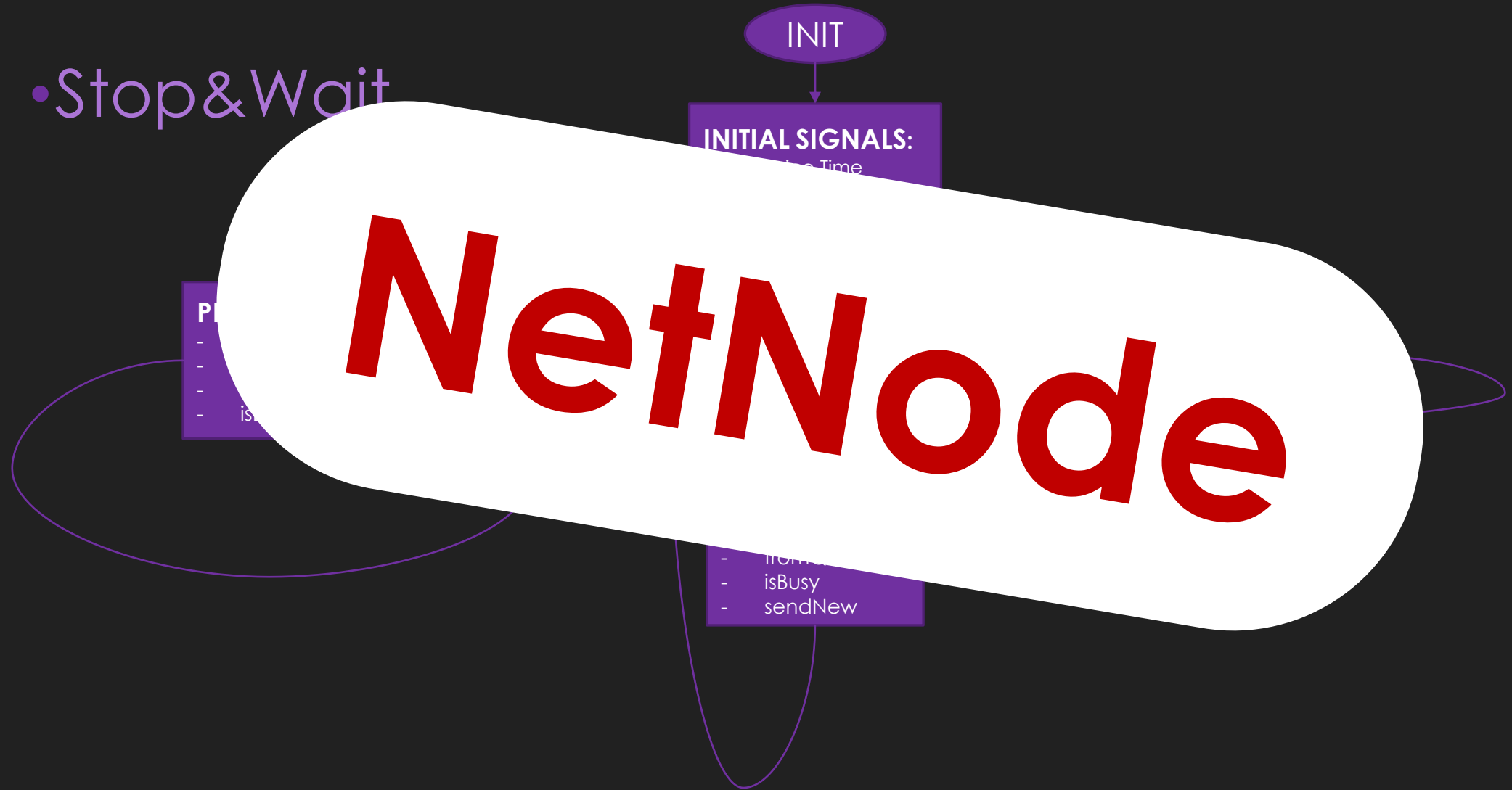
Máquina de estados

• Stop&Wait



Máquina de estados

- Stop&Wait



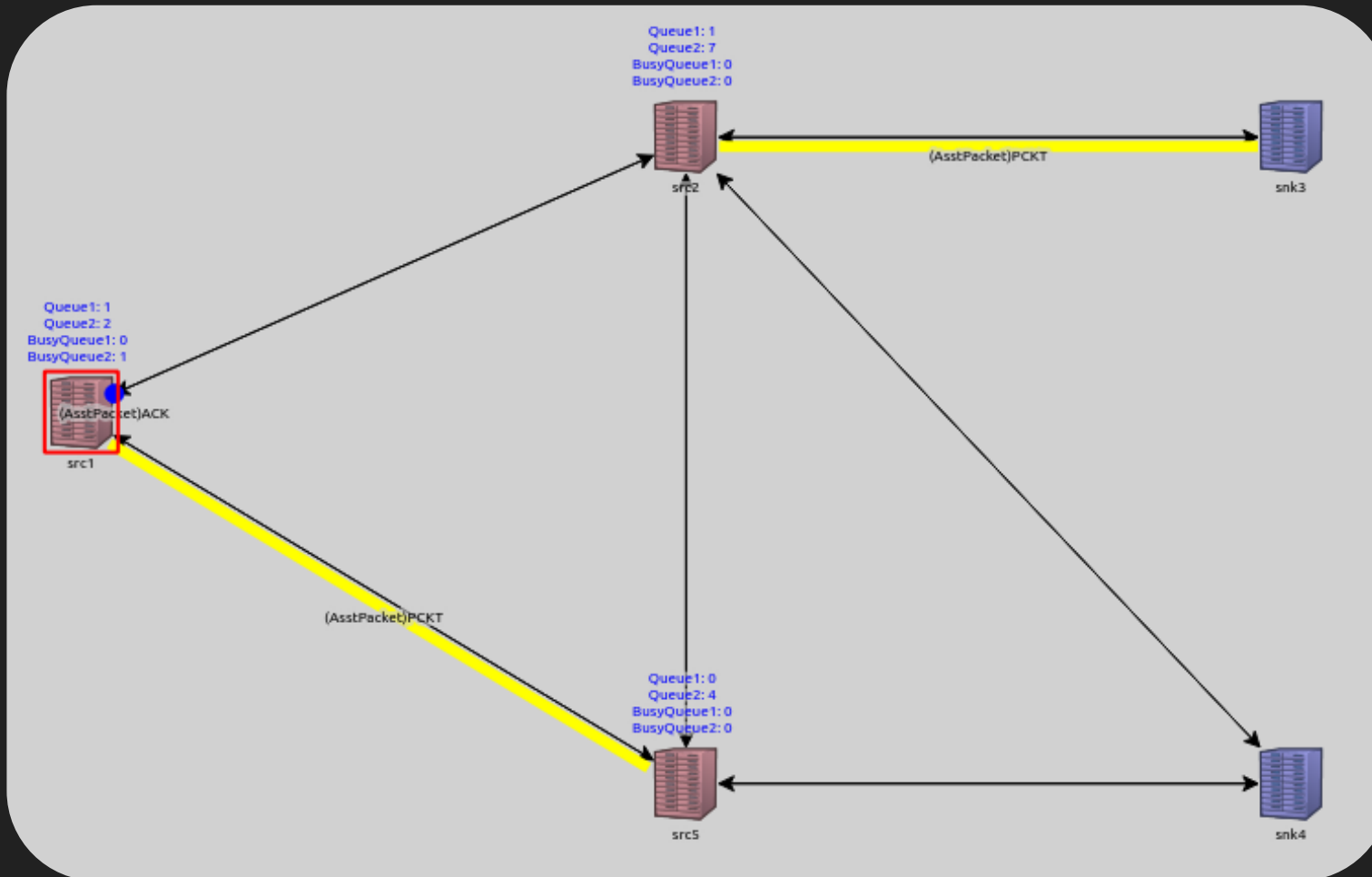
Stop & Wait

- Envío de paquetes (gestión de paquetes de fuente y reenviados)
- Gestión de errores (ACK's y NACK's)
- Retransmisiones en caso de error

```
rc = processMessage(packet, protocolType, &action);
if(rc != 0)
{
    EV << "ERROR: processMessage() \n";
    return;
}

rc = sendMessage(protocolType, action);
if(rc != 0)
{
    EV << "ERROR: sendMessage() \n";
    return;
}
```

Simulación

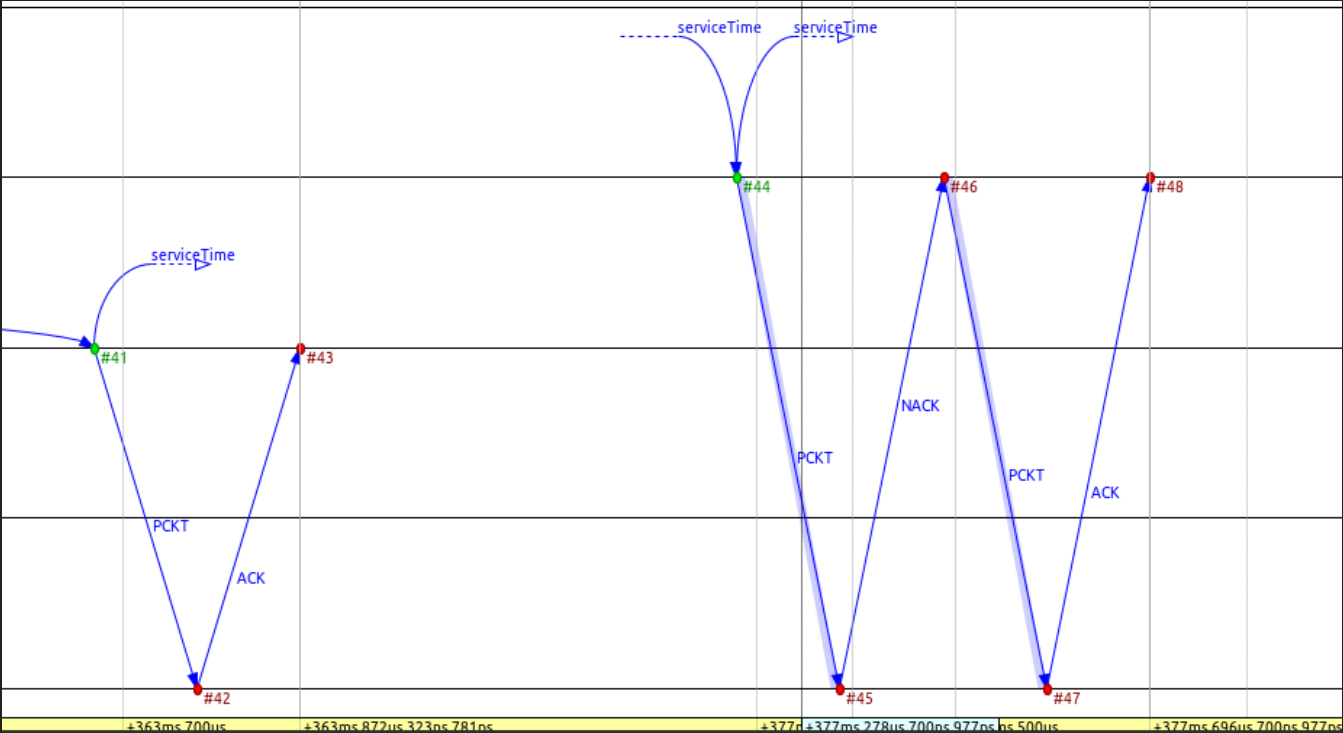


Simulación

```
INFO:Message type: 4 [fwd packet]
** Event #14 t=3.1905344892 RTT_CAS03_NET.src1 (FullNetNode, id=2) on ACK (asst_rtt_caso3::AsstPacket, id=157)
INFO:Message received from gate: 1
INFO:Message type: 2 [ACK]
** Event #15 t=3.421911284772 RTT_CAS03_NET.src5 (FullNetNode, id=4) on selfmsg serviceTime (asst_rtt_caso3::AsstPacket, id=46)
INFO:Message type: self-message
** Event #16 t=3.422388284772 RTT_CAS03_NET.snk4 (FullNetNode, id=0) on PKT (asst_rtt_caso3::AsstPacket, id=174)
INFO:Message type: 4 [fwd packet]
INFO:Packet[packet_from_src1_seq_10] Path[src1-src5-snk4] TotalTime[1.025]
** Event #17 t=3.422408384772 RTT_CAS03_NET.snk5 (FullNetNode, id=0) on ACK (asst_rtt_caso3::AsstPacket, id=181)
INFO:Message received from gate: 1
INFO:Message type: 2 [ACK]
** Event #18 t=3.719288424149 RTT_CAS03_NET.src2 (FullNetNode, id=3) on selfmsg arrivalTime (asst_rtt_caso3::AsstPacket, id=3)
** Event #19 t=3.883858935544 RTT_CAS03_NET.snk4 (FullNetNode, id=0) on selfmsg serviceTime (asst_rtt_caso3::AsstPacket, id=127)
INFO:Message type: self-message
** Event #20 t=3.983898071709 RTT_CAS03_NET.src5 (FullNetNode, id=4) on selfmsg arrivalTime (asst_rtt_caso3::AsstPacket, id=5)
** Event #21 t=4.952178988152 RTT_CAS03_NET.src2 (FullNetNode, id=3) on selfmsg serviceTime (asst_rtt_caso3::AsstPacket, id=140)
INFO:Message type: self-message
** Event #22 t=4.952958988152 RTT_CAS03_NET.snk4 (FullNetNode, id=0) on PKT (asst_rtt_caso3::AsstPacket, id=210)
INFO:Message type: 4 [fwd packet]
```

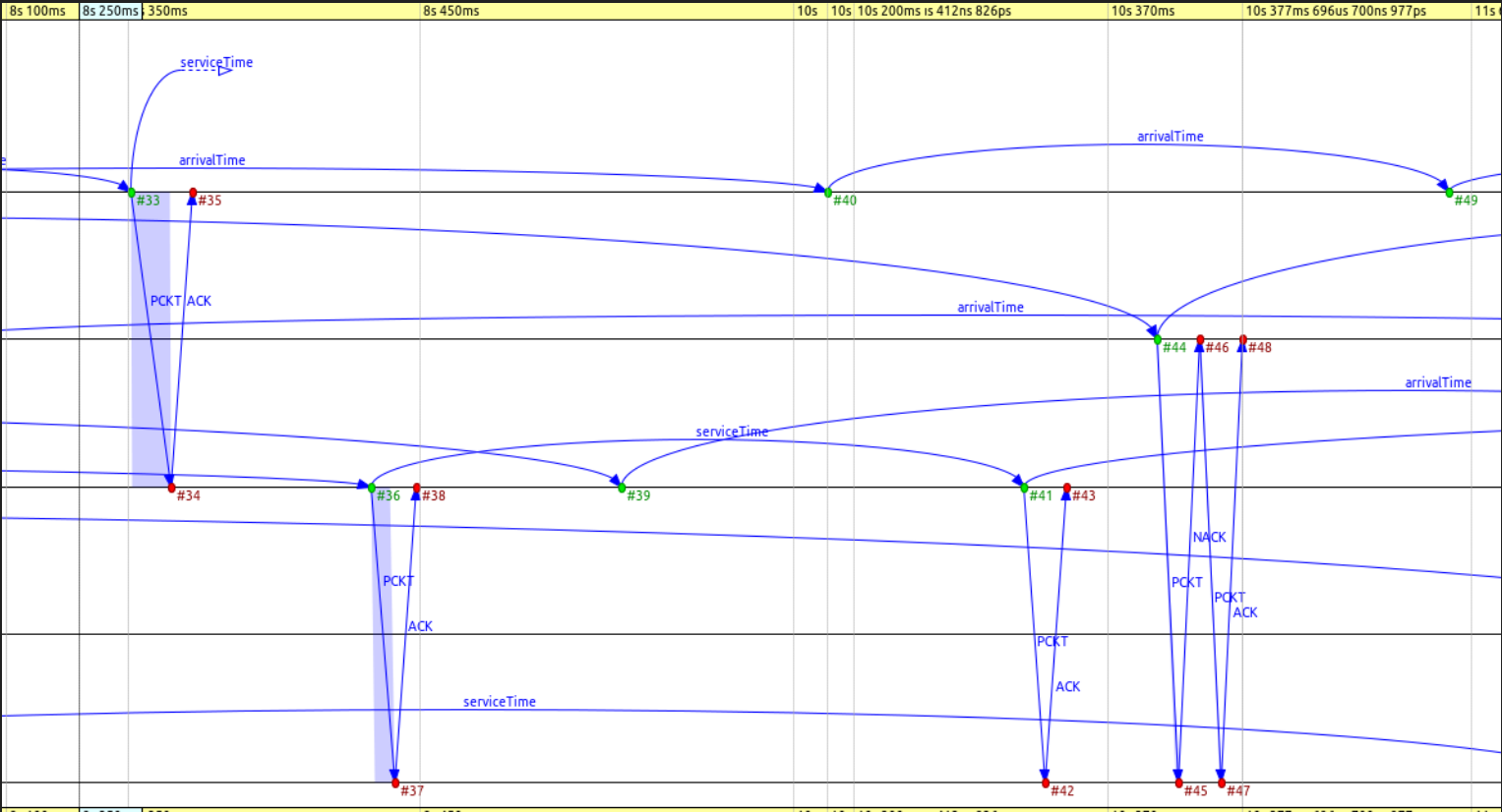
**Datos de paquetes
y red**

Resultados



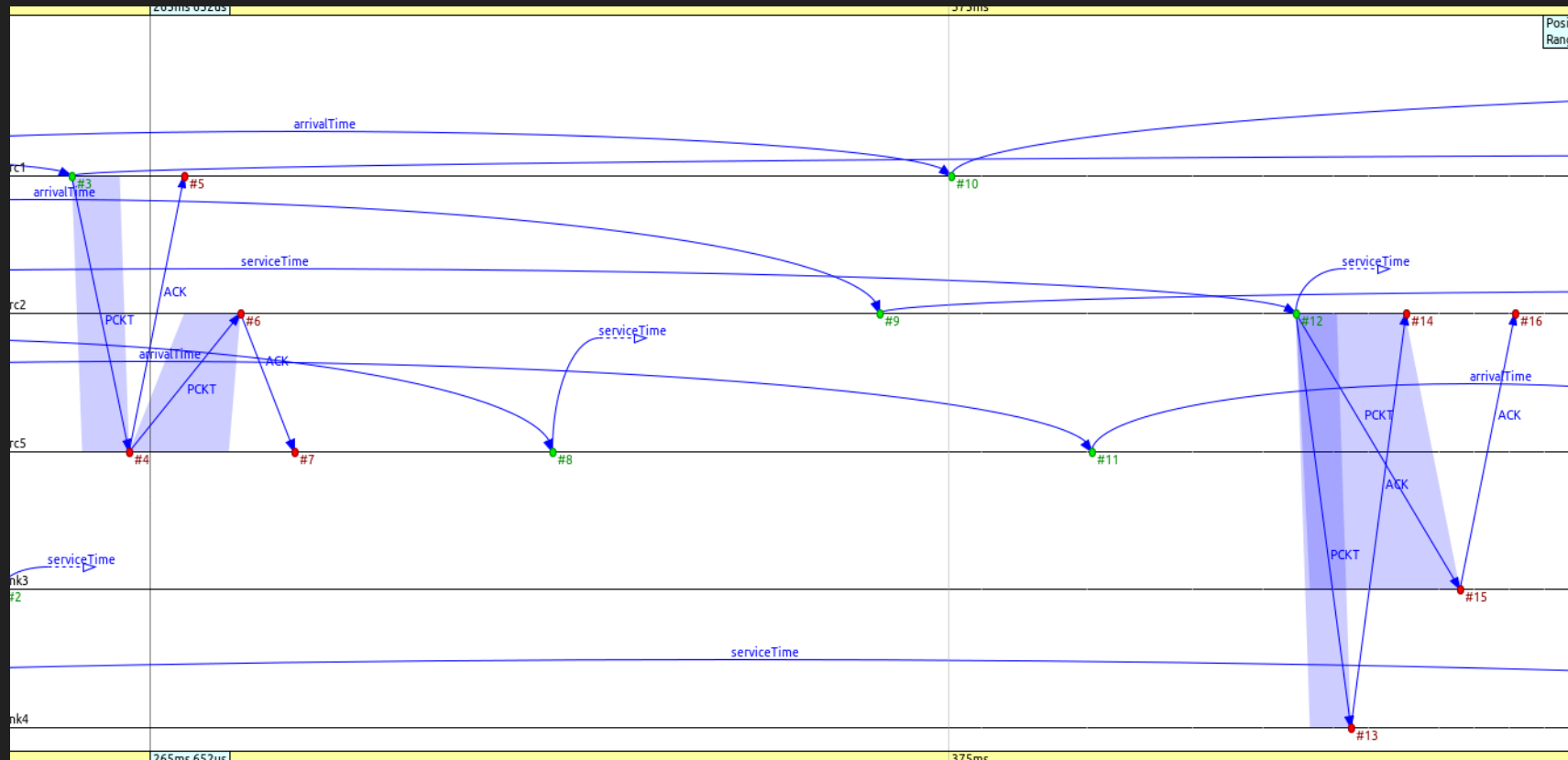
Protocolos

Resultados



Longitud exponencial

Resultados



Comunicaciones complejas y simultáneas

Conclusiones

- De gran utilidad para comprender mejor el funcionamiento de equipos, protocolos y comunicaciones de redes
- Lenguaje muy potente y versátil
- Complejidad de curva de aprendizaje en etapas iniciales.
- Documentación limitada
- Entorno de desarrollo útil, con opciones de debug y simulación bastante completas (interesante poder ver la pila de procesos e hilos para debug y errores)

Adrián Santiago Santo-Tomás

OMNet++ - Caso 3

Rendimiento en Redes de Telecomunicación
Curso 2020-2021

