

# Sistema de comunicaciones seguras con segmentación virtual de dominios

## Avance de proyecto

Alberto Daniel Lange

Dirección: Juan Ignacio Vaccarezza  
Codirección: Santiago Pérez Ghiglia

Ingeniería en Telecomunicaciones  
Instituto Balseiro

26 de febrero de 2025



## ① Introducción

## ② Revisión bibliográfica

## ③ Desarrollo

## ④ Conclusiones

# ① Introducción

## ② Revisión bibliográfica

## ③ Desarrollo

## ④ Conclusiones

# Contexto

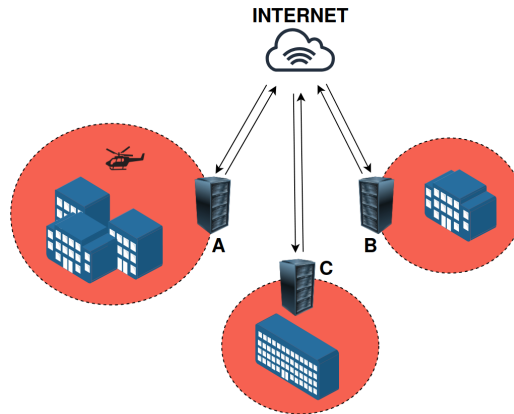


Figura 1: Esquema simplificado de operación del sistema.

# Progreso alcanzado anteriormente

- Definición y validación de arquitectura lógica en laboratorio virtual.

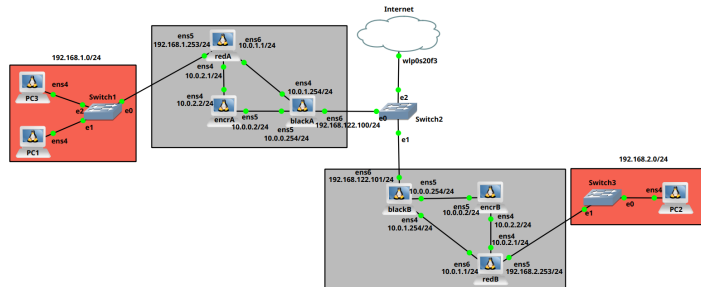


Figura 2: Simulación de la propuesta de solución en GNS3.

# Pendientes

- Validación del encriptador en ambiente virtualizado.
- Implementación sobre hardware.

① Introducción

② Revisión bibliográfica

③ Desarrollo

④ Conclusiones

## seL4

- Microkernel de código abierto.
- Hipervisor tipo 1.
- Aislamiento entre componentes garantizado.

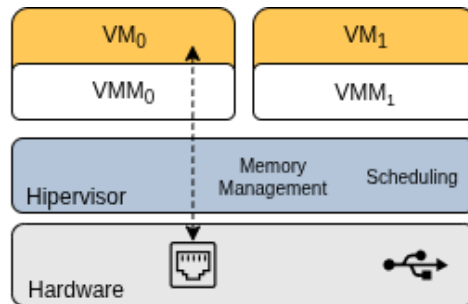


Figura 3: Esquema de entorno virtualizado usando seL4 como hipervisor.



- **Component Architecture for microkernel-based Embedded Systems.**
- Framework de desarrollo para seL4.
- Comunicación mediante interfaces bien definidas.

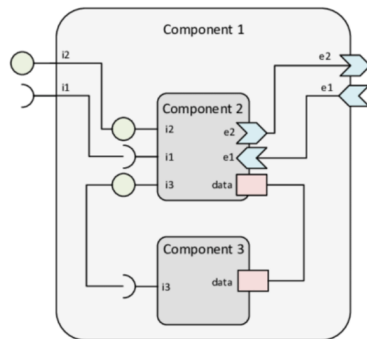


Figura 4: Ejemplo de componentes e interfaces en CamkES.

# Modelos ejemplo en CAmkES

## **minimal\_64**

- Implementación mínima de una VM para arquitectura x86\_64.
- Punto de partida para el desarrollo.

## **zmq\_samples**

- Ejemplo de comunicación entre tres VMs usando ZeroMQ.
- Modelo base para la implementación del encriptador.

① Introducción

② Revisión bibliográfica

**③ Desarrollo**

④ Conclusiones

# Estrategia de modelos en entornos virtualizados

## ¿Qué?

- Realizar modelos que validen progresivamente los componentes desarrollados.

## ¿Para qué?

- Ligar problemas concretos a cada modelo y resolverlos de forma independiente.
- Implementar un encriptador funcional en un entorno virtualizado como paso previo a trabajar con hardware.

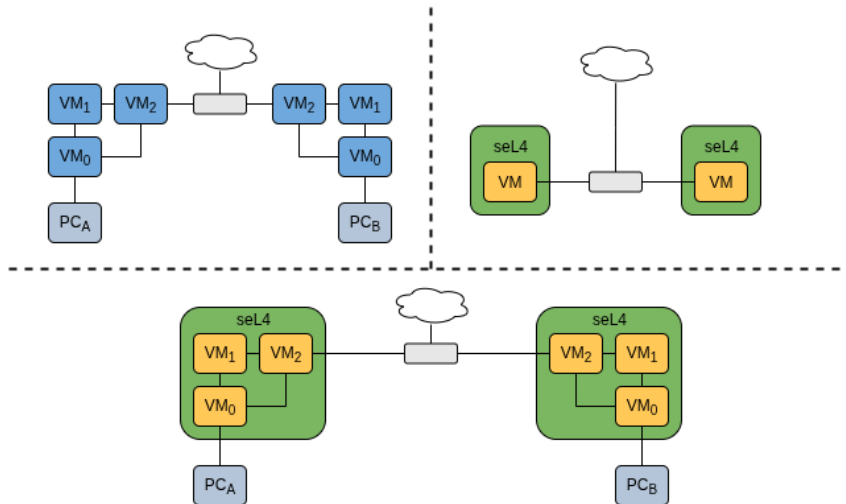


Figura 5: Modelos en entornos virtualizados.

# Modelo virtualizado

- Validar la arquitectura lógica del encriptador.
- Funcionalidad *split-tunneling*.

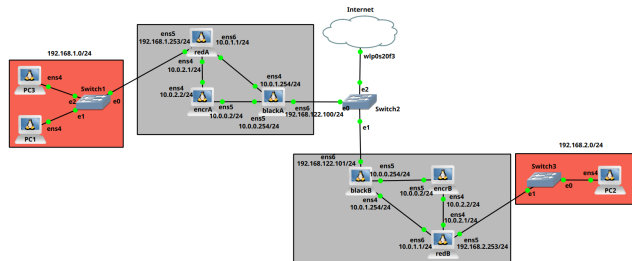


Figura 6: Simulación de la arquitectura lógica en GNS3.

# Modelo emulado

- Validación del passthrough de hardware y la compatibilidad del kernel Linux 4.9 con seL4.

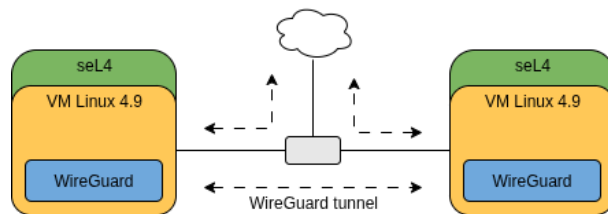


Figura 7: Esquema del modelo emulado.

## Modelo emulado: passthrough de hardware

CAMkES actúa de interfaz para la configuración del passthrough de dispositivos PCI a una VM a través de los Base Address Registers (BARs) e interrupciones.

### Ejemplo de configuración en CAMkES:

```
vm0.vm_ioports = [start, end];  
vm0.pci_devices = [bus, device, function, memory];  
vm0.vm_irqs = [source, dest];
```



## Modelo emulado: gestión de memoria VMs

`simple_untypedN_pool`: prealocación de memoria para las VMs.

`heap_size`: tamaño del heap del VMM.

`guest_ram_mb`: tamaño de la RAM asignada a cada VM.

# Modelo de hardware

- Solución completa como paso previo a la implementación en hardware.

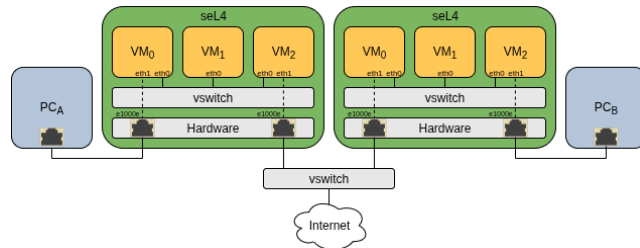


Figura 8: Esquema del modelo de hardware.

# Modelo de hardware: comunicación entre VMs

## ¿Cómo funciona?

### Virtio-Net en VM

- El host implementa un dispositivo PCI virtual (virtio-net).
- La VM encuentra este dispositivo y carga el driver correspondiente.
- En la transmisión, la VM escribe los datos a enviar en un buffer asociado a una cola del hipervisor. El VMM notifica al host que hay datos listos en la cola (virtqueue).

### Virtio-Net en host

- El host lee el paquete en la virtqueue y lo copia a un buffer dentro del espacio de memoria de la VM destino.
- El host inyecta una interrupción en la VM destino para notificarle que hay datos disponibles.

# Modelo de hardware: integración

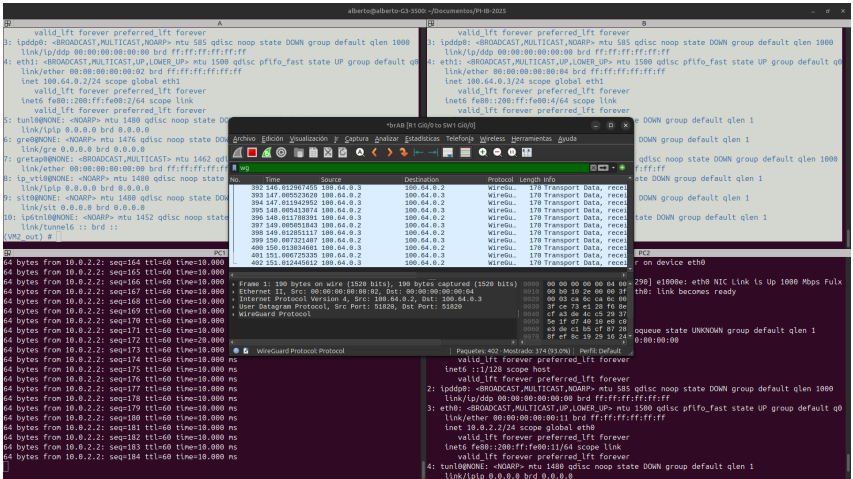


Figura 9: Comunicación entre PCs en nodos distintos.

The image displays a network configuration and monitoring session across three windows.

**Top Window (Terminal):** Shows the configuration of a bridge named 'br1'. The configuration includes setting the bridge to be a learning bridge, enabling the Spanning Tree Protocol (STP), and configuring the bridge to be a root bridge. The configuration is applied to the bridge interface 'br1'.

```

1: valid_lft forever preferred_lft forever
2: ipddp: <BROADCAST,MULTICAST,NOARP> mtu 585 qdisc noop state DOWN group default qlen 1000
   link/ipddp 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
   link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
   inet 100.64.0.2/24 scope global eth1
       valid_lft forever preferred_lft forever
   inet6 fe80::200:ff:fe00:a03/64 scope link
       valid_lft forever preferred_lft forever
4: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
   link/tunl0 0.0.0.0 brd 0.0.0.0
5: gre0@NONE: <NOARP> mtu 1476 qdisc noop state DOWN group default qlen 1000
   link/gre 0.0.0.0 brd 0.0.0.0
6: greap@NONE: <BROADCAST,MULTICAST> mtu 1462 qdisc noop state DOWN group default qlen 1000
   link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
7: ip_vti0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
   link/ipip 0.0.0.0 brd 0.0.0.0
8: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
   link/sit 0.0.0.0 brd 0.0.0.0
9: ip6ln@NONE: <NOARP> mtu 1452 qdisc noop state DOWN group default qlen 1000
   link/tunnel6 :: brd ::
   (VM2 out) #

```

**Middle Window (Wireshark):** Shows a packet capture of ICMP Echo (ping) requests and replies. The capture is filtered by 'icmp'. The packets are captured on the 'eth0' interface. The first packet is an ICMP Echo request from 100.64.0.2 to 100.64.0.2. The second packet is an ICMP Echo reply from 100.64.0.2 to 100.64.0.2.

No.	Time	Source	Destination	Protocol	Length	Info
595	267.041379308	8.8.8.8	100.64.0.2	ICMP	98	Echo (ping) request
596	268.013996068	100.64.0.2	8.8.8.8	ICMP	98	Echo (ping) reply
597	268.050596834	8.8.8.8	100.64.0.2	ICMP	98	Echo (ping) request
598	269.013616089	100.64.0.2	8.8.8.8	ICMP	98	Echo (ping) reply
599	269.041722993	8.8.8.8	100.64.0.2	ICMP	98	Echo (ping) request
600	270.01372006	100.64.0.2	8.8.8.8	ICMP	98	Echo (ping) reply
601	270.041137627	8.8.8.8	100.64.0.2	ICMP	98	Echo (ping) request
602	271.013145733	100.64.0.2	8.8.8.8	ICMP	98	Echo (ping) reply
603	271.040698657	8.8.8.8	100.64.0.2	ICMP	98	Echo (ping) request
604	272.012712739	100.64.0.2	8.8.8.8	ICMP	98	Echo (ping) reply
605	272.039579917	8.8.8.8	100.64.0.2	ICMP	98	Echo (ping) request

**Bottom Window (Terminal):** Shows the configuration of a bridge named 'br2'. The configuration includes setting the bridge to be a learning bridge, enabling the Spanning Tree Protocol (STP), and configuring the bridge to be a root bridge. The configuration is applied to the bridge interface 'br2'.

```

1: valid_lft forever preferred_lft forever
2: ipddp: <BROADCAST,MULTICAST,NOARP> mtu 585 qdisc noop state DOWN group default qlen 1000
   link/ipddp 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
   link/ether 00:00:00:00:00:11 brd ff:ff:ff:ff:ff:ff
   inet 10.0.2.2/24 scope global eth0
       valid_lft forever preferred_lft forever
   inet6 fe80::200:ff:fe00:a03/64 scope link
       valid_lft forever preferred_lft forever
4: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
   link/tunl0 0.0.0.0 brd 0.0.0.0

```

na de comunicaciones seguras con segmentación virtual de dominios

# Implementación en hardware



Figura 11: SuperMicro SYS-E300-9D.

## Características:

- CPU: Intel Xeon D-2123IT (4C/8T)
- RAM: 32 GB DDR4
- 4x NIC Intel i210 1GbE  
2x NIC Intel X553 10GbE
- Soporte para virtualización (VT-x, VT-d)

# Implementación en hardware: desafíos



Figura 12: SuperMicro SYS-E300-9D.

## Desafíos:

- ✓ Redirección de consola.
- ✗ Passthrough de controlador Ethernet.
- ✗ Throughput entre VMs.

## Implementación en hardware: throughput entre VMs

- La implementación actual requiere de 7 mapeos de memoria por paquete, lo que limita drásticamente el throughput.

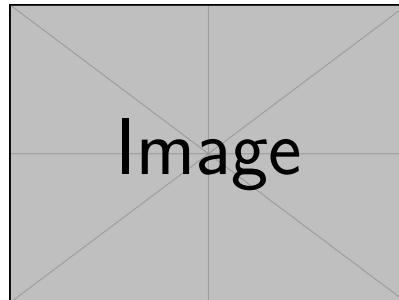


Figura 13: Throughput medido entre VMs.



# Implementación en hardware: throughput entre VMs

Existen alternativas para mejorar el rendimiento:

- Translation Vspace.
- Optimización de copias en memoria.

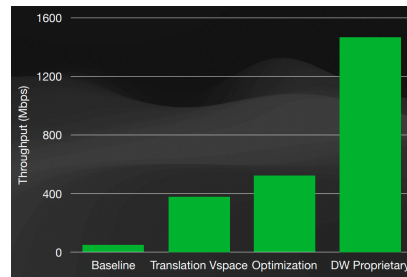


Figura 14: Optimizaciones logradas sobre el throughput entre VMs.  
Fuente: seL4 Summit 2023.

# Implementación en hardware: throughput entre VMs

Existen alternativas para mejorar el rendimiento:

- Translation Vspace.
- Optimización de copias en memoria.

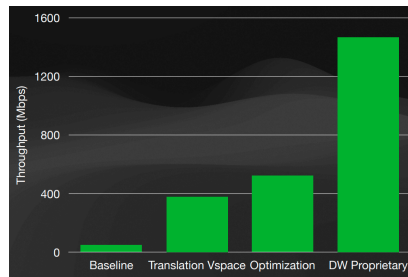
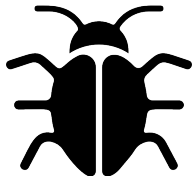


Figura 15: Optimizaciones logradas sobre el throughput entre VMs.  
Fuente: seL4 Summit 2023.

- 1 Introducción
- 2 Revisión bibliográfica
- 3 Desarrollo
- 4 Conclusiones**

# Resumen

- Introducción a los modelos de trabajo utilizados.
- Validación del encriptador en un entorno virtualizado.
- Avances en la implementación sobre hardware.

## Trabajo a futuro

- Resolver desafíos pendientes en la implementación de hardware.
- Documentar y presentar la solución final.

¡Muchas gracias!  
¿Preguntas?