Network Services on Service Extensible Routers *

Lukas Ruf
Computer Engineering and Networks Laboratory (TIK)
Swiss Federal Institute of Technology (ETH) Zürich

IWAN 2005, Sophia-Antipolis. 22. November 2005

*Generously supported by IBM Zürich Rüschlikon, Intel Corp. Grant 22919

Motivation

Two major trends in networking:

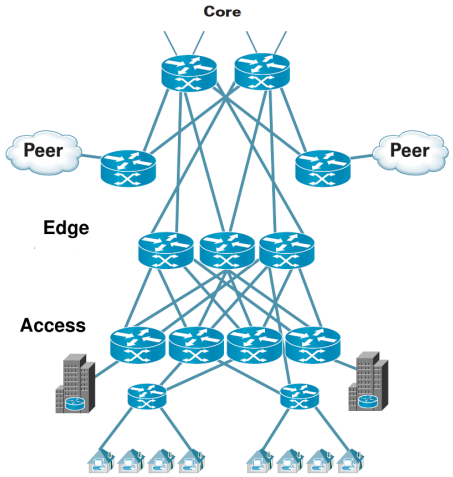
- Function shift
  - concentration of functions on routers instead of end systems
  - platform for new emerging services

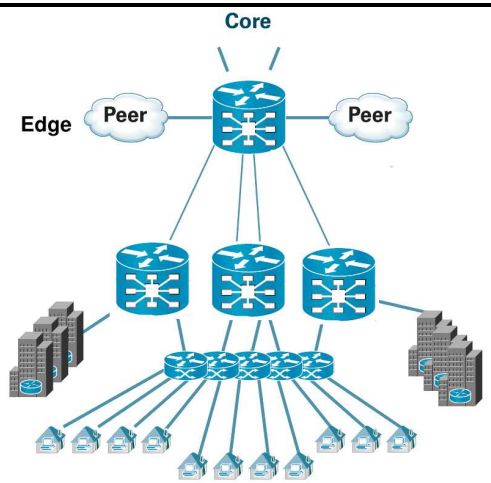
Motivation

Two major trends in networking:

- Function shift
  - concentration of functions on routers instead of end systems
  - platform for new emerging services
- Network simplification
  - router consolidation
  - less but larger router devices

Today	Tomorrow



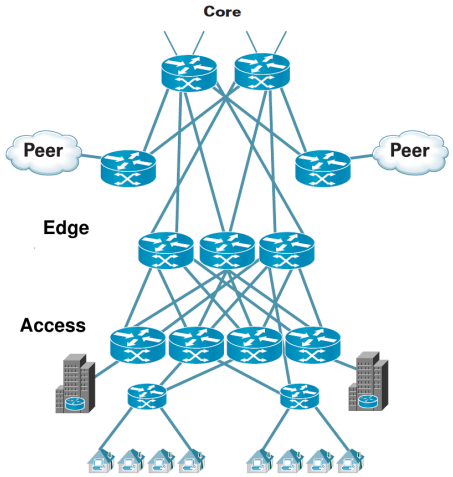


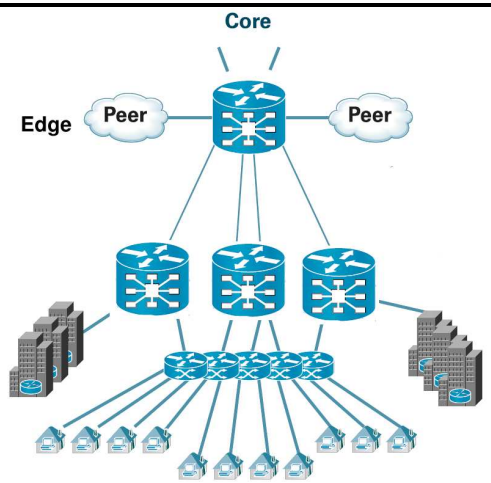
Motivation

Two major trends in networking:

- Function shift
  - concentration of functions on routers instead of end systems
  - platform for new emerging services
- Network simplification
  - router consolidation
  - less but larger router devices

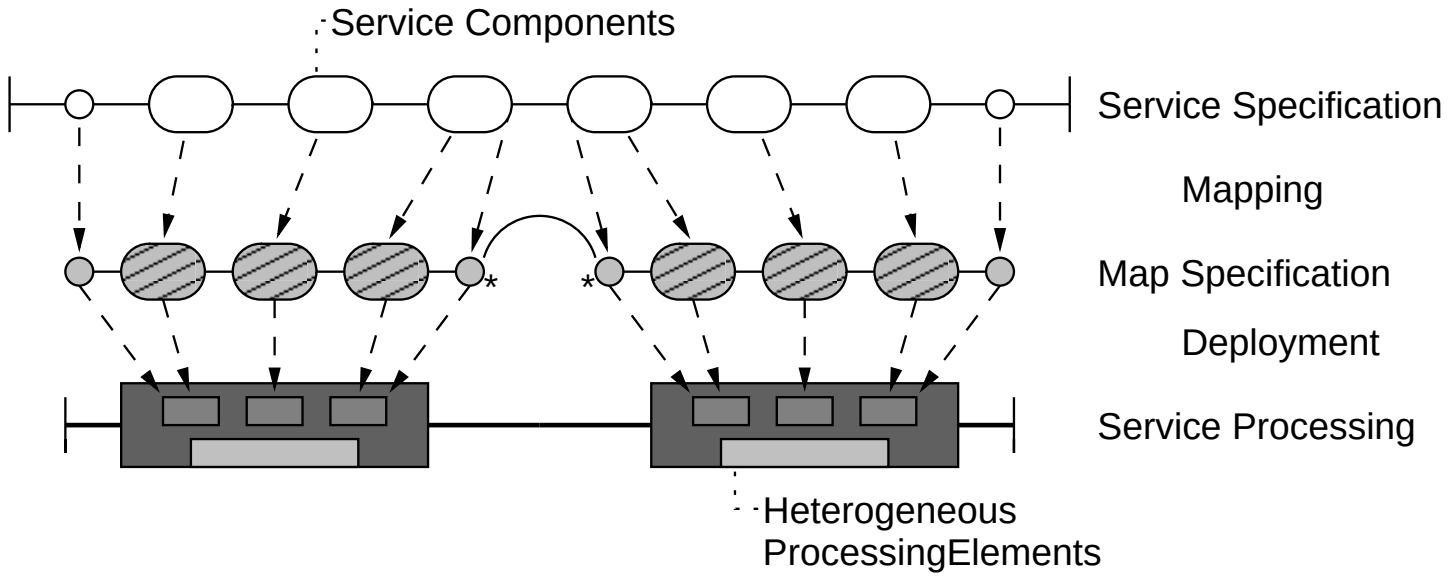
Today	Tomorrow





Required: Flexible service infrastructure for future high-performance routers supporting service creation on demand.

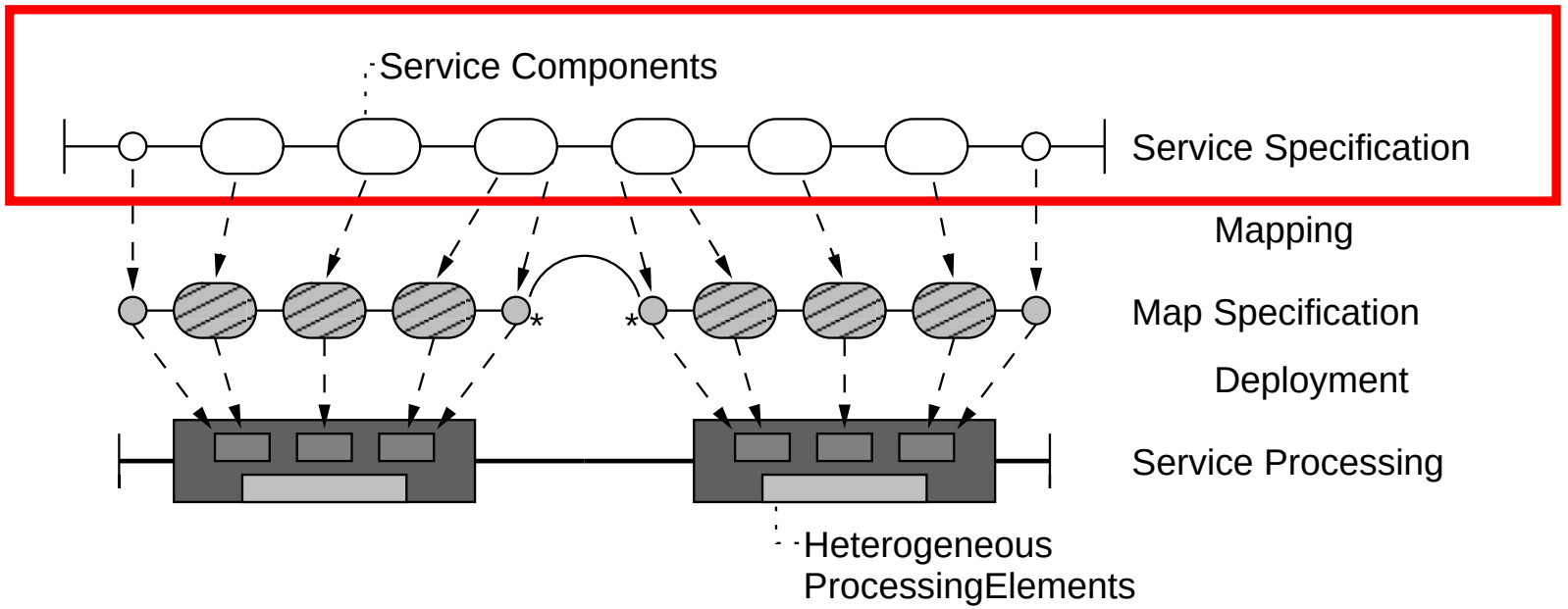
The Big Picture



Key Parts

- ☐ Service model
- ☐ Service programming interface
- ☐ Service mapping algorithm
- ☐ Non-disruptive service creation
- ☐ Service processing infrastructure
- ☐ Router management and control

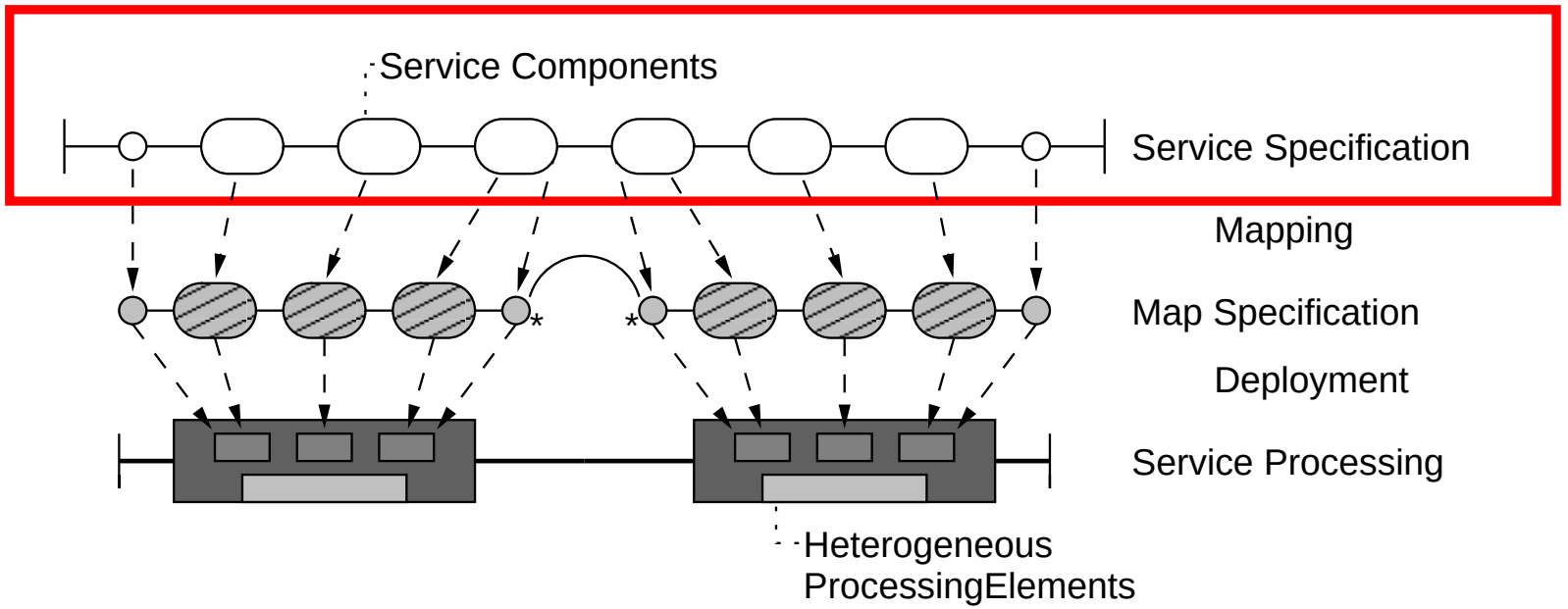
Focus of this presentation



Key Parts

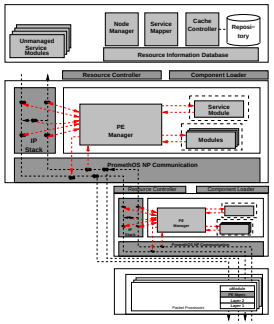
- ☐ **Service model**
- ☐ **Service programming interface**
- ☐ Service mapping algorithm
- ☐ Non-disruptive service creation
- ☐ Service processing infrastructure
- ☐ Router management and control

Focus of this presentation



Key Parts

- ☐ **Service model**
- ☐ **Service programming interface**
- ☐ Service mapping algorithm
- ☐ Non-disruptive service creation
- ☐ Service processing infrastructure
- ☐ Router management and control



Goals and Requirements

Service Model

G: Describe component-based network services
G: Define a service infrastructure
R: Programmability of data and control plane
R: Service internal control mechanisms
R: Programmable classification mechanisms
R: Flexible extensibility
R: Slim, efficient

Goals and Requirements

Service Model

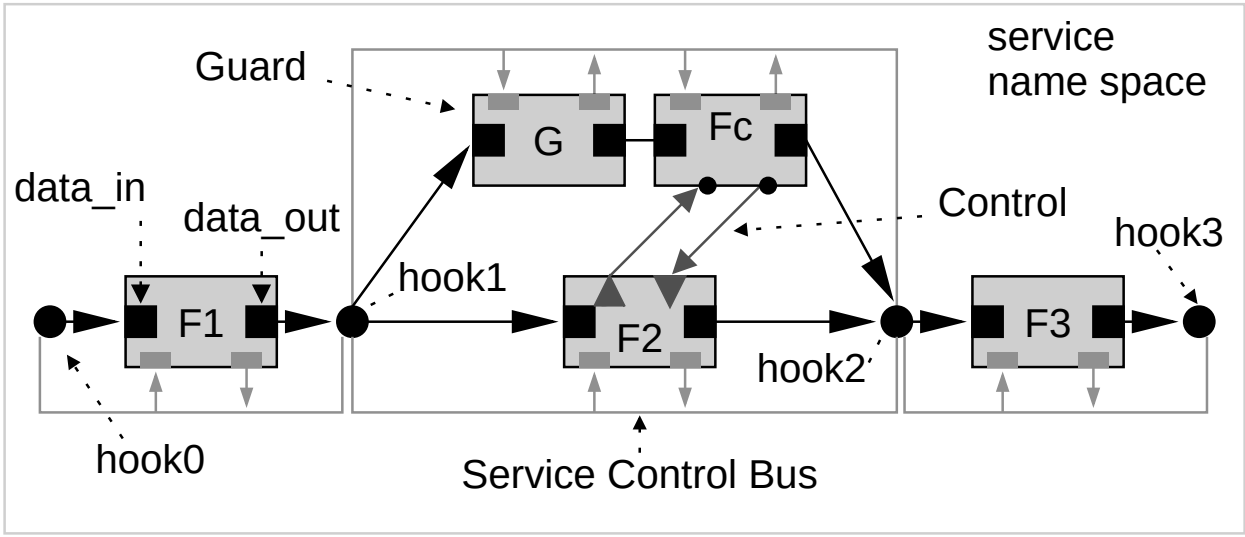
G: Describe component-based network services
G: Define a service infrastructure
R: Programmability of data and control plane
R: Service internal control mechanisms
R: Programmable classification mechanisms
R: Flexible extensibility
R: Slim, efficient

Service Programming Interface

G: Specification of network services
R: Coping with flexibility of our service model
R: Definition of resource constraints
R: Efficiently processable

Service Model

Graph of Service Components

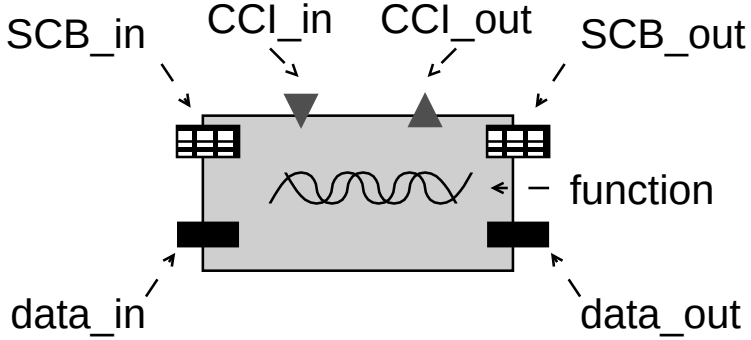


Constituent Elements

- Data path service components
- Control service components
- Service chains: staged, pipelined data path processing
- Guards: packet classification
- Hooks: dynamically created extension places
- Name spaces: service encapsulation
- Service control bus: signalling between components

Data Path Service Component

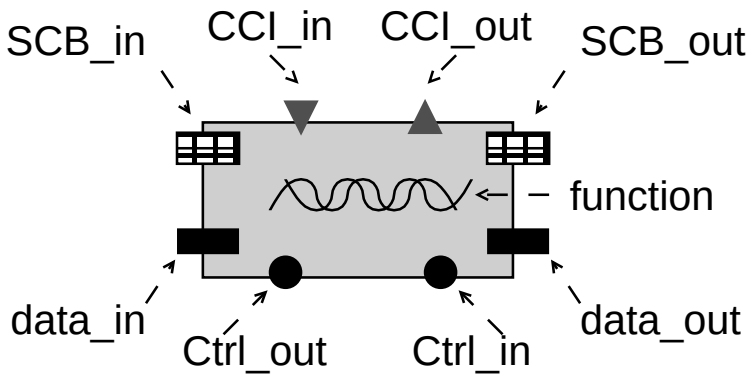
Network Traffic Processing Functions



- Function: according to the plugin model
- Data Path I/O: data_in and data_out
- Service Control Bus: SCB_in and SCB_out
- Component Control Interface: CCI_in and CCI_out (optional)

Control Service Component

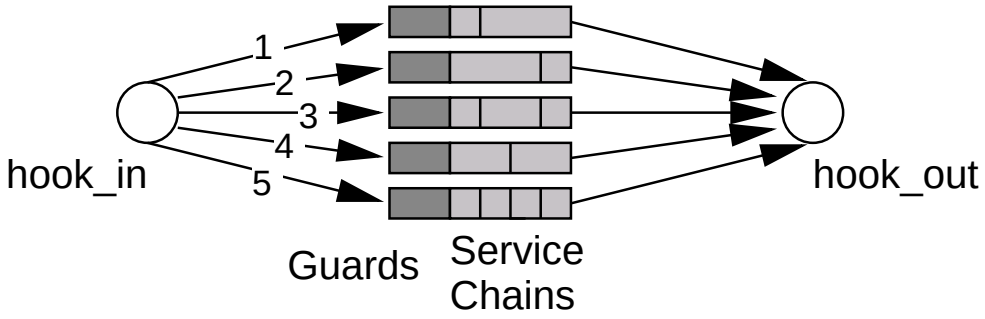
Service Internal Control Functions



- Function: according to the plugin model
- Data Path I/O: data_in and data_out
- Service Control Bus: SCB_in and SCB_out
- Component Control Interface: CCI_in and CCI_out (optional)
- **Controller Interface**: Ctrl_in and Ctrl_out

Extension and Dispatching

Data Path Service Extension and Packet Dispatching



- Hooks
  - Created on demand
  - Dispatching per service processing environment
- Guards
  - Classification
- Service Chains
  - Pipelined data path service components
  - Embedded between pairs of hooks
- Packet dispatching
  - *First-match-first-consume* and *copy* semantics

Service Programming Interface

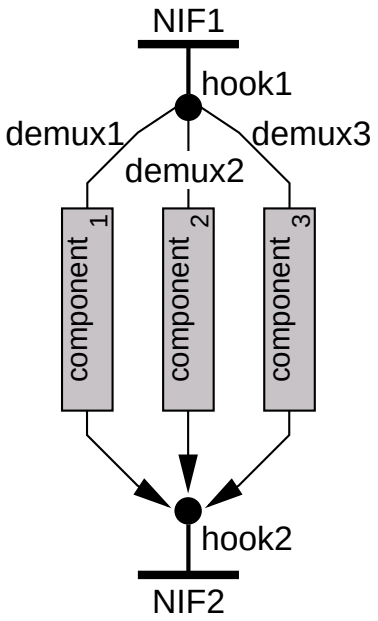
The Service Programming Language

```
ID           = "#" VALID_NAME.
TIMED        = "timed="DELAY.
BW_RES       = "bwmin="BW "bwmax="BW [ "pps="NUMBER ].
CPU_RES      = "cpumin="CPU "cpumax="CPU.
RAM_RES      = "type="ID "rammin="RAM "rammax="RAM.
PROC_TYPE    = ("ia32"|"ia64"|"np4"|"np4_pp"|"ixp2400"|"ixp2400_pp"|....).
CTRL_INFO    = (STRING | "file=" VALID_NAME ).
COMP_SPEC    = ( "src" [ ID ] | "bin" ( PROC_TYPE | ID ) )
               [ "|" CPU_RES ] [ { "|"  RAM_RES } ].
COMP_IDENT   = ( [ "(" COMP_SPEC ")" ] VALID_NAME ID | ID ).
SERV_COMP    = COMP_IDENT [ ":" ID ] "(" [ CTRL_INFO ] ")".
CTRL_COMP    = [ TIMED ] SERV_COMP { "!" ID "@"NUMBER }.
CTRL_CHAIN   = "{" { CTRL_COMP } "}".
COMP_STRING= "{" { SERV_COMP } "}".
GUARD        = "[" [ "|" BW_RES ] [ SERV_COMP ] "]".
HOOK_IN      = (ID | ">" ID [ "copy" ]   "?" INTF).
HOOK_OUT     = (ID | ">" ID [ "copy" ] [ "?" INTF ]).
SERV_CHAIN   = HOOK_IN
                "@" [ TIMED ] [ GUARD ] COMP_STRING "@"
               HOOK_OUT.
SERVICE      = "{" ID [ "!"  CTRL_CHAIN ] { SERV_CHAIN } "}".
```

- Formal language to specify network services
- Defined by an extended EBNF
- Linear, concise specification of service graphs
- Definition of important constraints

Exemplary Service: Three Parallel Service Chains

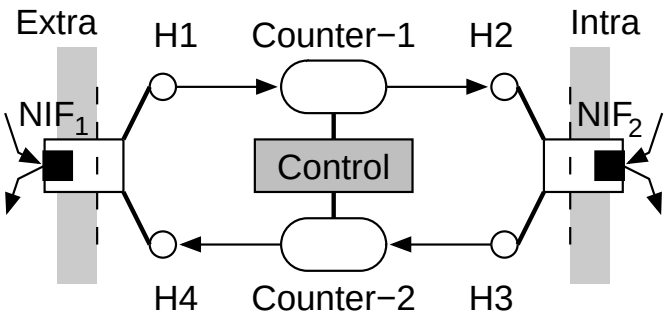
Visualization	Chain 1	Chain 2	Chain 3

	`{ #threeparallel`
	`> #hook1`
	`? NIF1`
	`@/* HOOK */`
	`[ /*DEMUX1*/ ]`
	`{ /*COMP_STRING*/`
	`( bin ia32)`
	`component1`
	`#instance1ID`
	`(/*CTRL_INFO*/)`
	`}`
	`@/* HOOK */`
	`> #hook2`
	`? NIF2`
		`/*extend hook1*/`
		`#hook1`
		
		`@/* HOOK */`
		`[/*DEMUX2*/ ]`
		`{/*COMP_STRING*/`
		`(bin ia32 )`
		`component2`
		`#instance2ID`
		`(/*CTRL_INFO*/)`
		`}`
		`@/* HOOK */`
		`#hook2`
			`/*extend hook1*/`
			`#hook1`
			
			`@/* HOOK */`
			`[ /*DEMUX3*/ ]`
			`{ /*COMP_STRING*/`
			`(bin ia32)`
			`component3`
			`#instance3ID`
			`(/*CTRL_INFO*/)`
			`}`
			`@/* HOOK */`
			`#hook2`
			`}/*Service End*/`

- ☐ Creation of hooks
- ☐ Component instance identification
- ☐ Service chain embedding
- ☐ Resource specification
- ☐ Binding to network interfaces

Exemplary Service: Packet Counting

Counting with Control



- Component control interface
- Control interface multiplexing
- 1:N control relation

Service Program

```
{#counter_service /* service ID*/
 !{ /* Control Chain              */
   =1s /* Interval                */
   (bin ia32) /* Specification    */
   Control.ko /* Binary Code      */
   #ControlID /* InstanceID       */
   ("reset") /* Ctrl Info         */
   : #ControlInterface /* CCI     */
   ! #Counter1Ctrl@1 /* CCI       */
   ! #Counter2Ctrl@2 /* CCI       */
 } /* end of Control Chain        */
 /* Counter−1 Service Chain       */
 > #H1          /* create hook     */
 copy           /* copy method     */
 ? NIF1         /* from NIF1       */
 @ { /* Component String          */
   (src)        /* Specification   */
   Counter.c /* Source Code        */
   #Counter1ID  /* InstanceID      */
   : #Counter1Ctrl /* CCI         */
   ("reset") /* Ctrl Info         */
 } /* end of Component String     */
 @ > #H2        /* create hook     */
 copy           /* copy method     */
 ? NIF2         /* NIF2            */
 /* Counter−2 Service Chain       */
 > #H3          /* create hook     */
 copy           /* copy method     */
 ? NIF2         /* from NIF2       */
 @ { /* Component String          */
   (src)        /* Specification   */
   Counter.c  /* Source Code       */
   #Counter2ID /* Instance ID      */
   : #Counter2Ctrl /* CCI         */
   ("reset")  /* Ctrl Info         */
 } /* end of Component String     */
 @ > #H4        /* create hook     */
 copy           /* copy method     */
 ? NIF1         /* to NIF1         */
}
```

Summary And Conclusion

Service Model

- Graph of service components
- Data path and control service components
- Service internal control
- Hooks: created on demand
- Service chains: efficient, staged data plane processing
- Service control bus: Powerful signalling method
- Guards: programmable classification mechanisms

Summary And Conclusion

Service Model

- Graph of service components
- Data path and control service components
- Service internal control
- Hooks: created on demand
- Service chains: efficient, staged data plane processing
- Service control bus: Powerful signalling method
- Guards: programmable classification mechanisms

Defines the service infrastructure of PromethOS NP

Summary And Conclusion

Service Model

- Graph of service components
- Data path and control service components
- Service internal control
- Hooks: created on demand
- Service chains: efficient, staged data plane processing
- Service control bus: Powerful signalling method
- Guards: programmable classification mechanisms

Defines the service infrastructure of PromethOS NP

Service Programming Language

- Copes with the flexibility of our service model
- Syntax defined by a modified EBNF
- Fast processable

Summary And Conclusion

Service Model

- Graph of service components
- Data path and control service components
- Service internal control
- Hooks: created on demand
- Service chains: efficient, staged data plane processing
- Service control bus: Powerful signalling method
- Guards: programmable classification mechanisms

Defines the service infrastructure of PromethOS NP

Service Programming Language

- Copes with the flexibility of our service model
- Syntax defined by a modified EBNF
- Fast processable

Provides the service programming interface of PromethOS NP

Thanks

Questions?

Lukas Ruf, <[lukas.]ruf@([[tik.]ee.]ethz|lpr).ch> :−)