lecture 20:
Header Space Analysis —
Static Checking For Networks
5590: software defined networking

anduo wang, Temple University
TTLMAN 401B, R 17:30-20:00

HSA

header space

- general and protocol agnostic
  - extend to new protocols and new types of checks (?)

statically check

- reachability properties
  - reachability failures, forwarding loops, traffic isolation and leakage

evaluation

- verify reachability between two subnets in 13 seconds

Peyman Kazemian., et al. "Header space analysis: static checking for networks"

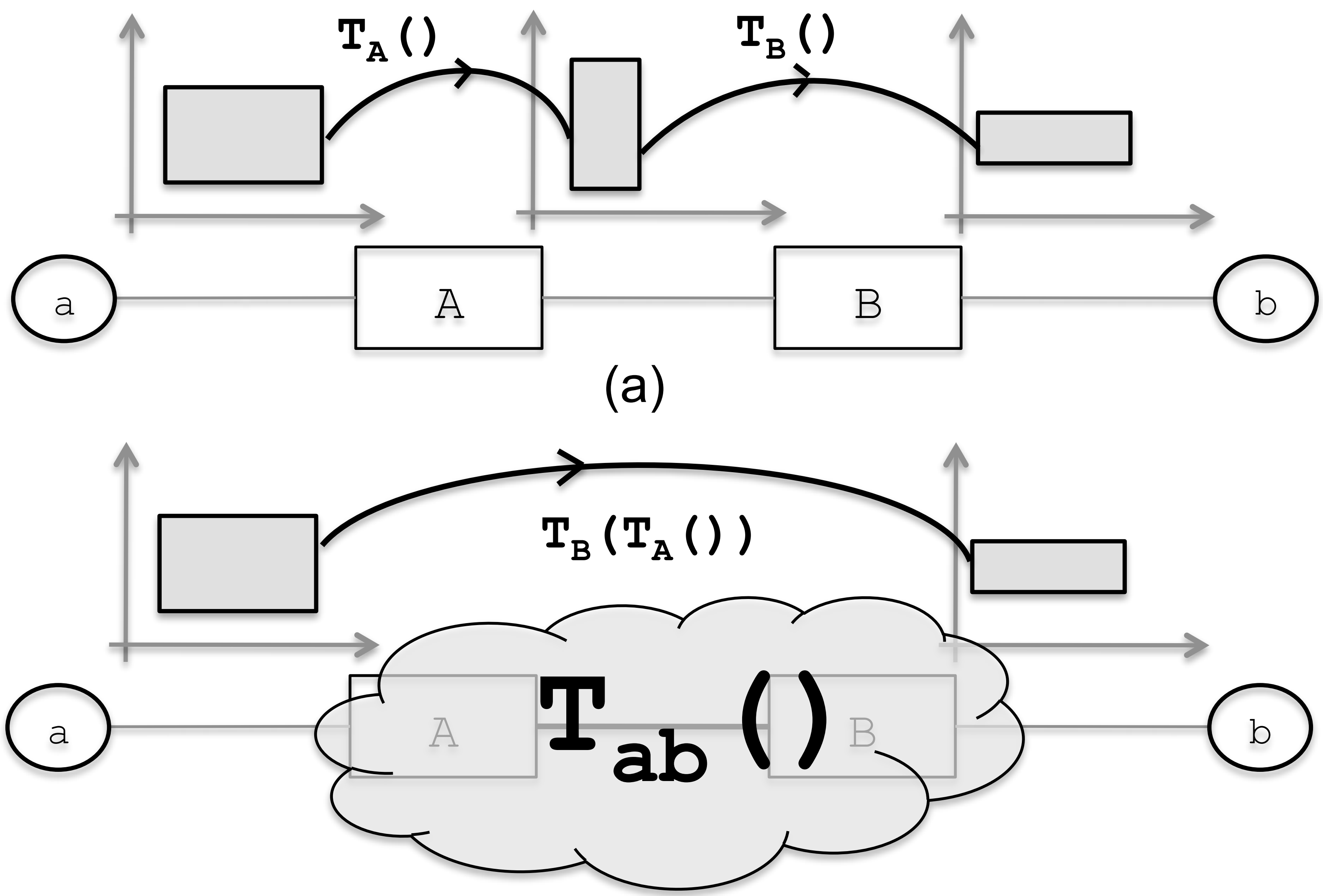
discussion (motivation)

debugging reachability is very time consuming

- complexity of the network state

HSA helps?

header space abstraction



header space abstraction

header space *H*

- $\{0,1\}^L$, where L is the header length
- a wildcard expression
  - sequence of L bits of 0, 1, or x (wildcard)
  - a region in header space: union of wildcard expressions

network space N

- $\{0,1\}^L \times \{1,\ldots,P\}$, where $\{1,\ldots,P\}$ is the list of ports

network transfer function

- a node transfer function $T: (h,p) \rightarrow \{(h_1, p_1), (h_2, p_2), \ldots\}$

header space abstraction

network transfer function

- a node transfer function T: (h,p) → {(h_1, p_1), (h_2, p_2), …}
- network transfer function

$$\Psi(h,p) = \begin{cases} T_1(h,p) & \text{if } p \in switch_1 \\ \dots & \dots \\ T_n(h,p) & \text{if } p \in switch_n \end{cases}$$

- topology transfer function

$$\Gamma(h,p) = \begin{cases} \{(h,p^*)\} & \text{if } p \text{ connected to } p^* \\ \{\} & \text{if } p \text{ is not connected.} \end{cases}$$

- multi-hop packet traversal

$$\Psi(\Gamma(\dots(\Psi(\Gamma(h,p)\dots)$$

using header space abstraction

an IPv4 router that forwards subnet S_1 traffic to port p_1, S_2 traffic to p_2, and S_3 traffic to p_3

$$T_r(h,p) = \begin{cases} \{(h,p_1)\} & \text{if } ip\_dst(h) \in S_1 \\ \{(h,p_2)\} & \text{if } ip\_dst(h) \in S_2 \\ \{(h,p_3)\} & \text{if } ip\_dst(h) \in S_3 \\ \{\} & \textit{otherwise.} \end{cases}$$

set operation on *H*

header space algebra

- determine how different spaces overlap
- basic set operation
  - *intersection, union, complementation, difference*

set operation on H — *intersection*

b_i \ b'_i	0	1	x
0	0	z	0
1	z	1	1
x	0	1	x

examples

$$11000\text{xxx} \cap \text{xx}00010\text{x} = 1100010\text{x}$$

$$1100\text{xxxx} \cap 111001\text{xx} = 11\text{z}001\text{xx} = \phi$$

set operation on *H* — *union*

cannot be simplified

example

- 1111xxxx and 0000xxxx

algorithm for logic minimization

- 10xx U 011x reduced to $b_4\overline{b_3} \oplus \overline{b_4}b_3b_2$

set operation on H — *complementation*

$h' \leftarrow \phi$
for bit b_i in h **do**
  if $b_i \neq x$ **then**
    $h' \leftarrow h' \cup$ x...x$\overline{b_i}$x...x
  end if
end for
return h'

set operation on H — *complementation*

algorithm for computing complement for h:

$h' \leftarrow \phi$
for bit b_i in h **do**
  if $b_i \neq x$ **then**
    $h' \leftarrow h' \cup$ x...x$\overline{b_i}$x...x
  end if
end for
return h'

set operation on H — *complementation*

algorithm for computing complement for h:

$h' \leftarrow \phi$
for bit b_i in h **do**
  if $b_i \neq x$ **then**
    $h' \leftarrow h' \cup$ x...x$\overline{b_i}$x...x
  end if
end for
return h'

set operation on H — *complementation*

algorithm for computing complement for h:

$h' \leftarrow \phi$
for bit b_i in h **do**
  if $b_i \neq x$ **then**
    $h' \leftarrow h' \cup$ x...x$\overline{b_i}$x...x
  end if
end for
return h'

set operation on H — *complementation*

algorithm for computing complement for h:

$h' \leftarrow \phi$
for bit b_i in h **do**
  if $b_i \neq x$ **then**
    $h' \leftarrow h' \cup$ x...x$\overline{b_i}$x...x
  end if
end for
return h'

set operation on H — *complementation*

algorithm for computing complement for h:

$h' \leftarrow \phi$
for bit b_i in h **do**
  if $b_i \neq x$ **then**
    $h' \leftarrow h' \cup$ x...x$\overline{b_i}$x...x
  end if
end for
return h'

set operation on H — *complementation*

algorithm for computing complement for h:

$h' \leftarrow \phi$
for bit b_i in h **do**
  if $b_i \neq x$ **then**
    $h' \leftarrow h' \cup$ x...x$\overline{b_i}$x...x
  end if
end for
return h'

set operation on H — *complementation*

algorithm for computing complement for h:

$h' \leftarrow \phi$
for bit b_i in h **do**
  if $b_i \neq x$ **then**
    $h' \leftarrow h' \cup$ x...x$\overline{b_i}$x...x
  end if
end for
return h'

example

set operation on H — *complementation*

algorithm for computing complement for h:

$h' \leftarrow \phi$
for bit b_i in h **do**
  if $b_i \neq x$ **then**
    $h' \leftarrow h' \cup$ x...x$\overline{b_i}$x...x
  end if
end for
return h'

example

$$(100\text{xxxxx})' = 0\text{xxxxxxx} \cup \text{x}1\text{xxxxxx} \cup \text{xx}1\text{xxxxx}$$

set operation on *H* — *difference*

$A - B = A \cap B'$. For example:

$$\begin{aligned}
&100\text{xxxxx} - 10011\text{xxx} = \\
&100\text{xxxxx} \cap (0\text{xxxxxxx} \cup \text{x}1\text{xxxxxx} \cup \text{xx}1\text{xxxxx} \\
&\cup \text{xxx}0\text{xxxx} \cup \text{xxxx}0\text{xxx}) \\
&= \phi \cup \phi \cup \phi \cup 1000\text{xxxx} \cup 100\text{x}0\text{xxx} \\
&= 1000\text{xxxx} \cup 100\text{x}0\text{xxx}.
\end{aligned}$$

header space analysis — reachability

can packets from host a reach host b

$$R_{a \to b} = \bigcup_{a \to b \text{ paths}} \{T_n(\Gamma(T_{n-1}(......(\Gamma(T_1(h,p)...))\}$$

header space analysis — reachability

can packets from host a reach host b

$$R_{a \to b} = \bigcup_{a \to b \text{ paths}} \{T_n(\Gamma(T_{n-1}(......(\Gamma(T_1(h,p)...))\}$$

range reverse

If header $h \subset \mathcal{H}$ reached b along the $a \to S_1 \to ... \to S_{n-1} \to S_n \to b$ path, then the original header sent by a is:

$$h_a = T_1^{-1}(\Gamma(...(T_{n-1}^{-1}(\Gamma(T_n^{-1}((h,b))...)),$$

using the fact that $\Gamma = \Gamma^{-1}$.

header space analysis — reachability

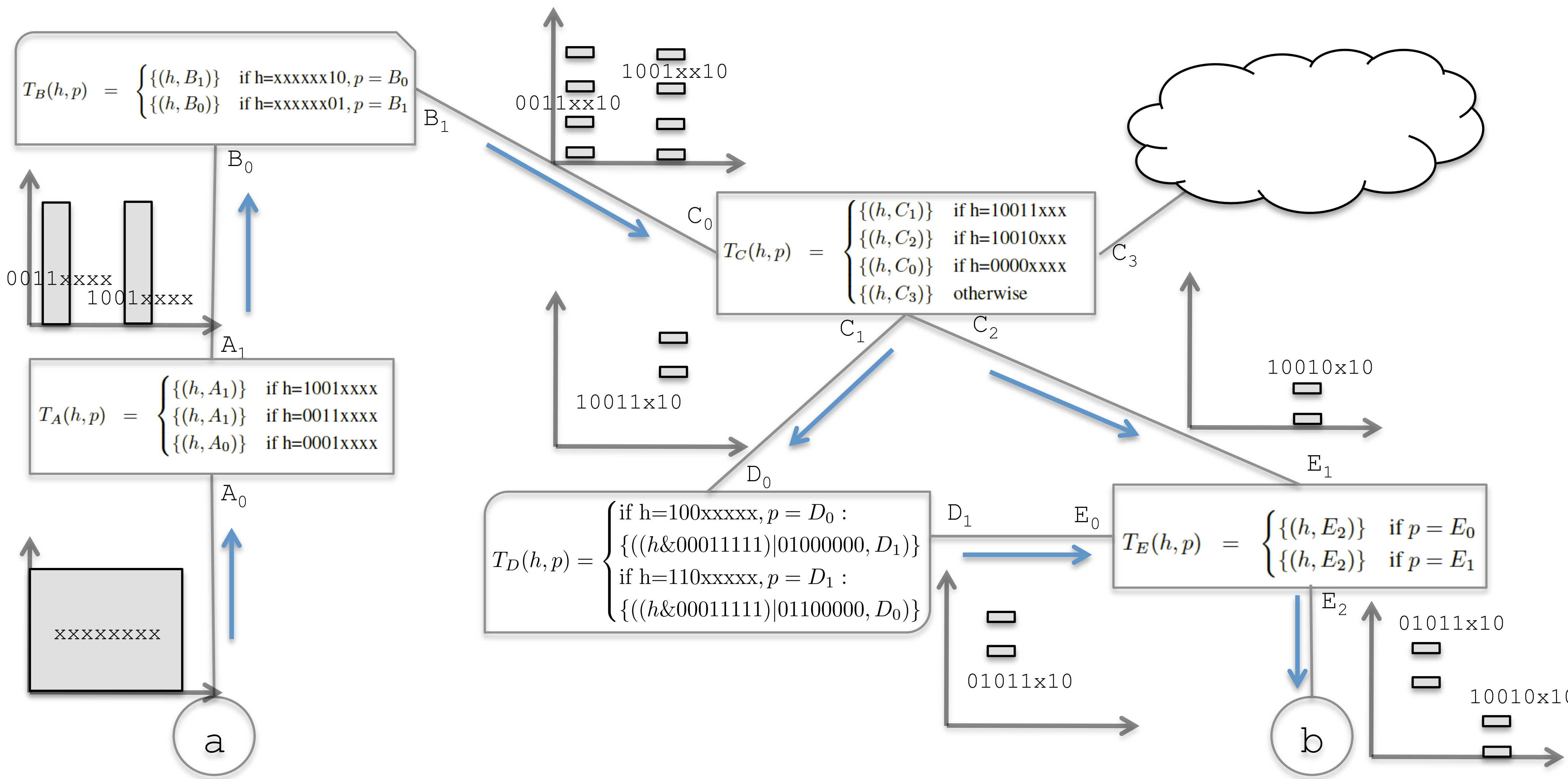


Figure 2: Example for computing reachability function from a to b. For simplicity, we assume a header length of 8 and show the first 4 bits on the x-axis and the last 4 bits on the y-axis. We show the range (output) of each transfer function composition along the paths that connect a to b. At the end, the packet headers that b will see from a are 01011x10 ∪ 10010x10.

header space analysis — reachability

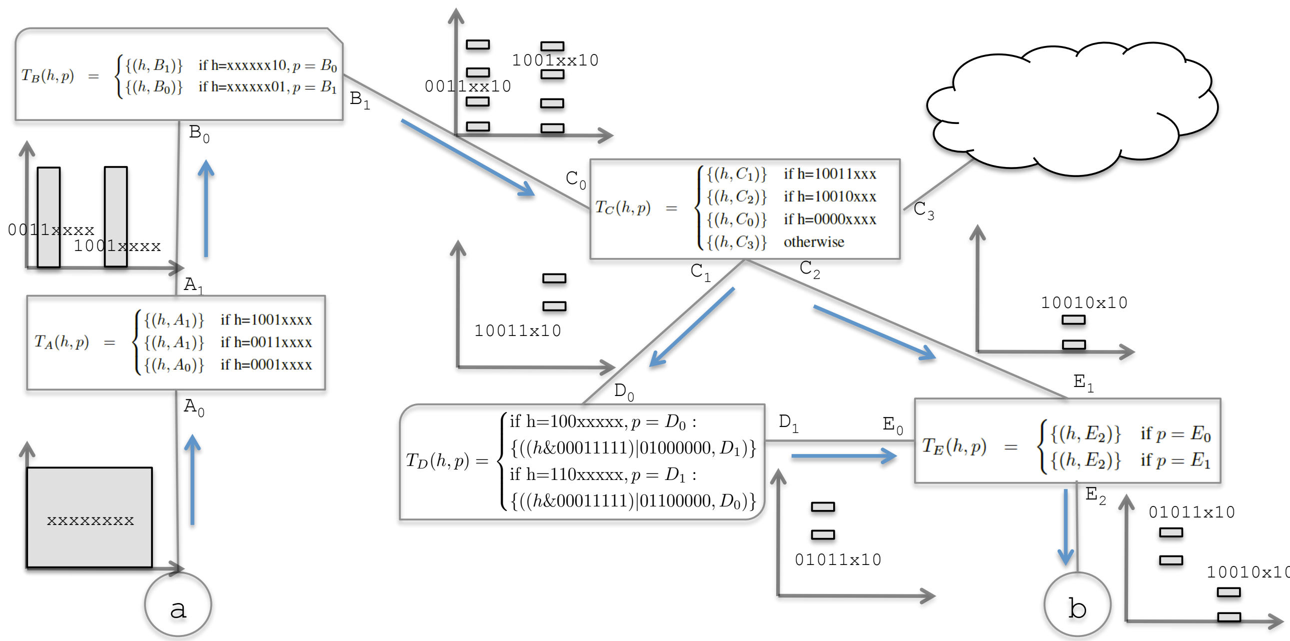


Figure 2: Example for computing reachability function from a to b. For simplicity, we assume a header length of 8 and show the first 4 bits on the x-axis and the last 4 bits on the y-axis. We show the range (output) of each transfer function composition along the paths that connect a to b. At the end, the packet headers that b will see from a are 01011x10 ∪ 10010x10.

$$R_{a\to b}(h,p) = \begin{cases}\text{if h=10010x10}, p = A_0 :\\ \{(h, E_2)\}\\ \text{if h=10011x10}, p = A_0 :\\ \{((h\&00011111)|01000000, E_2)\}\end{cases}$$

header space analysis — reachability

worst case complexity

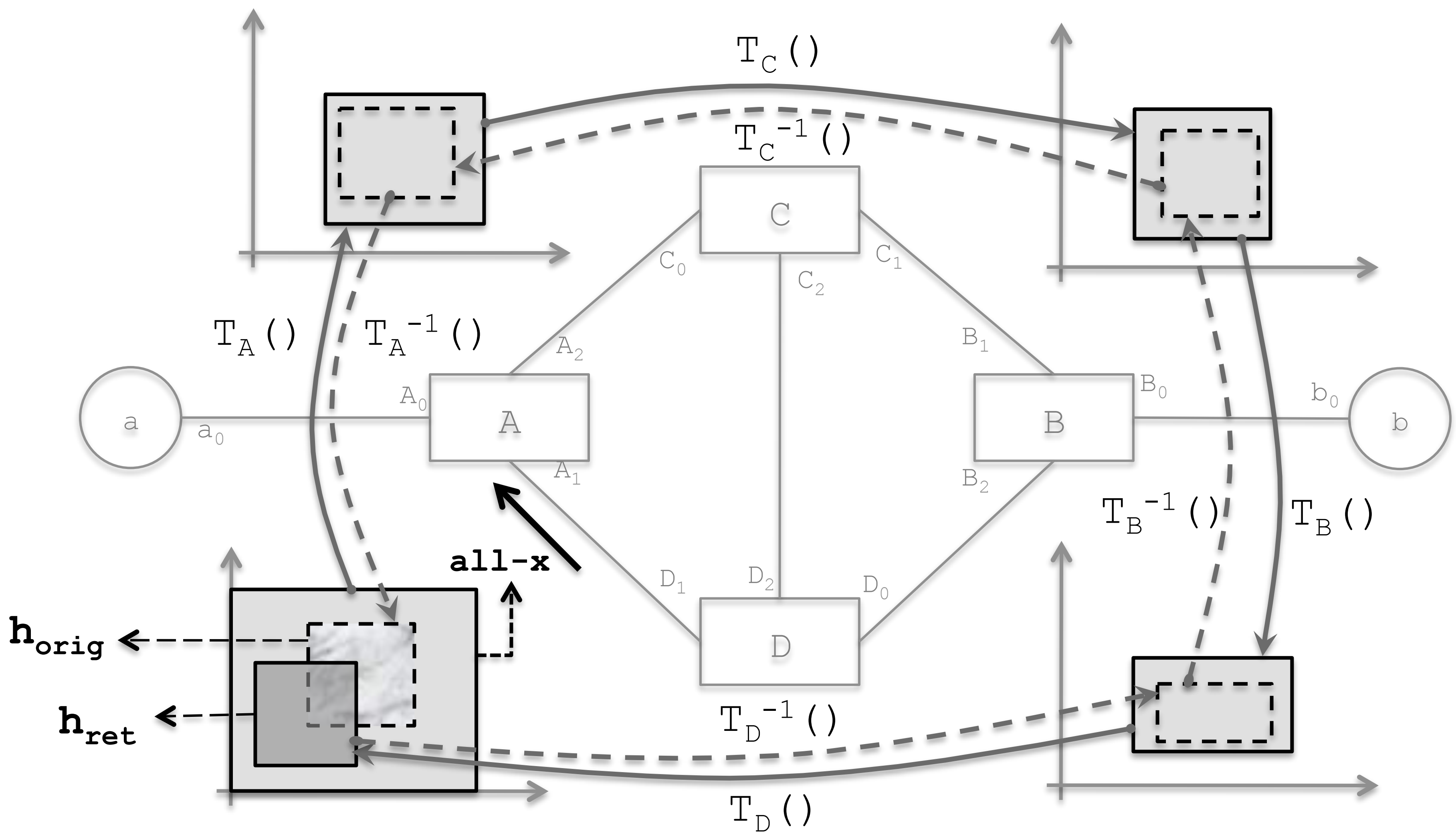
- assume input of
  - R_1 wildcard expressions, R_2 transfer function rules
- output $O(R_1R_2)$ wildcard expressions

linear fragmentation assumption

- as packet propagates to the core of the network, the match pattern will be less specific
- cR rather than R^2 where $c << R$
- running time becomes $O(dR^2)$
  - d is the network diameter
  - R is the maximum number of forwarding rules in a router

brute force: $O(2^L)$

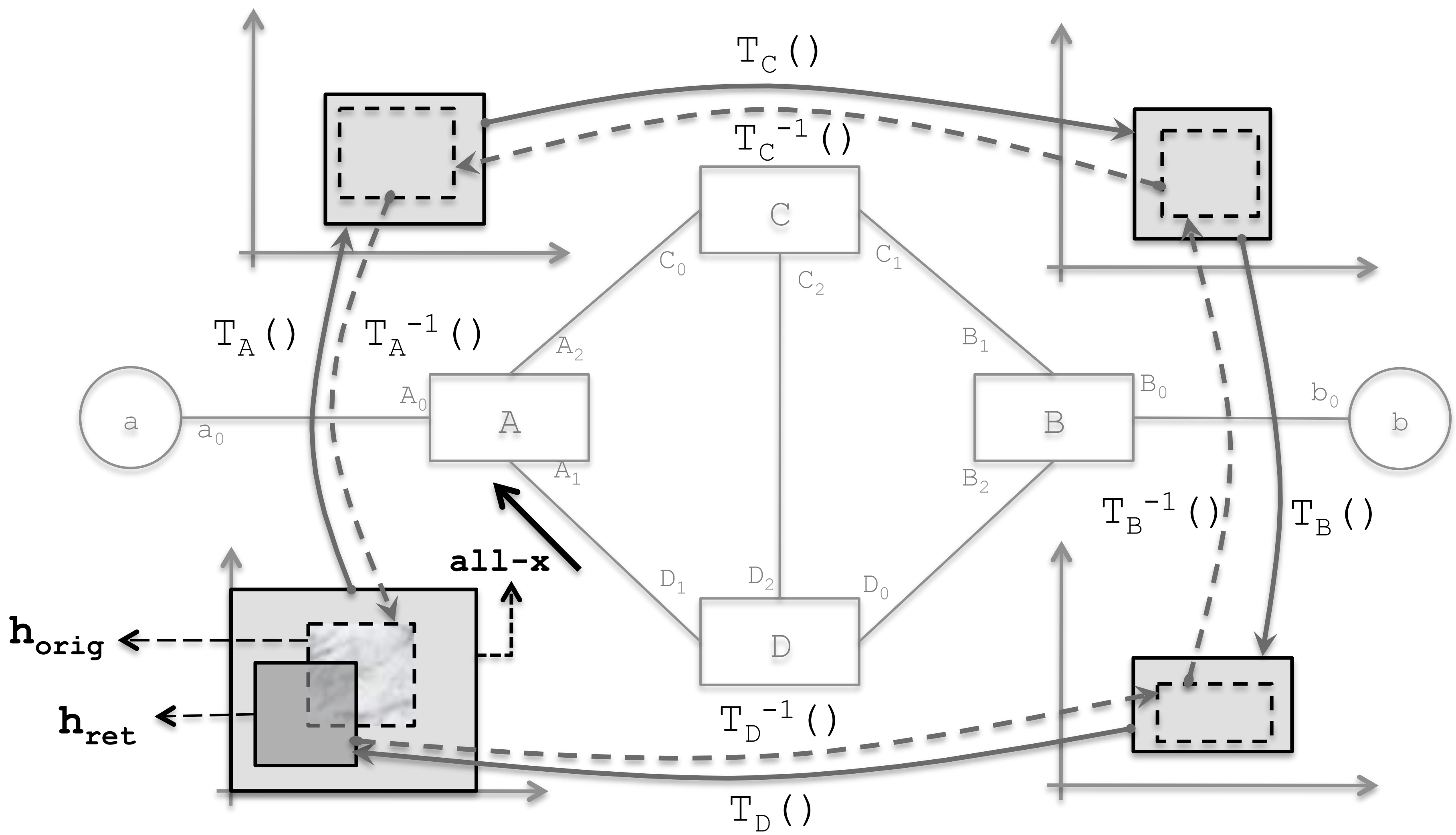
header space analysis — loop



catch loop

- inject an all-x test packet header from each port and track the packet

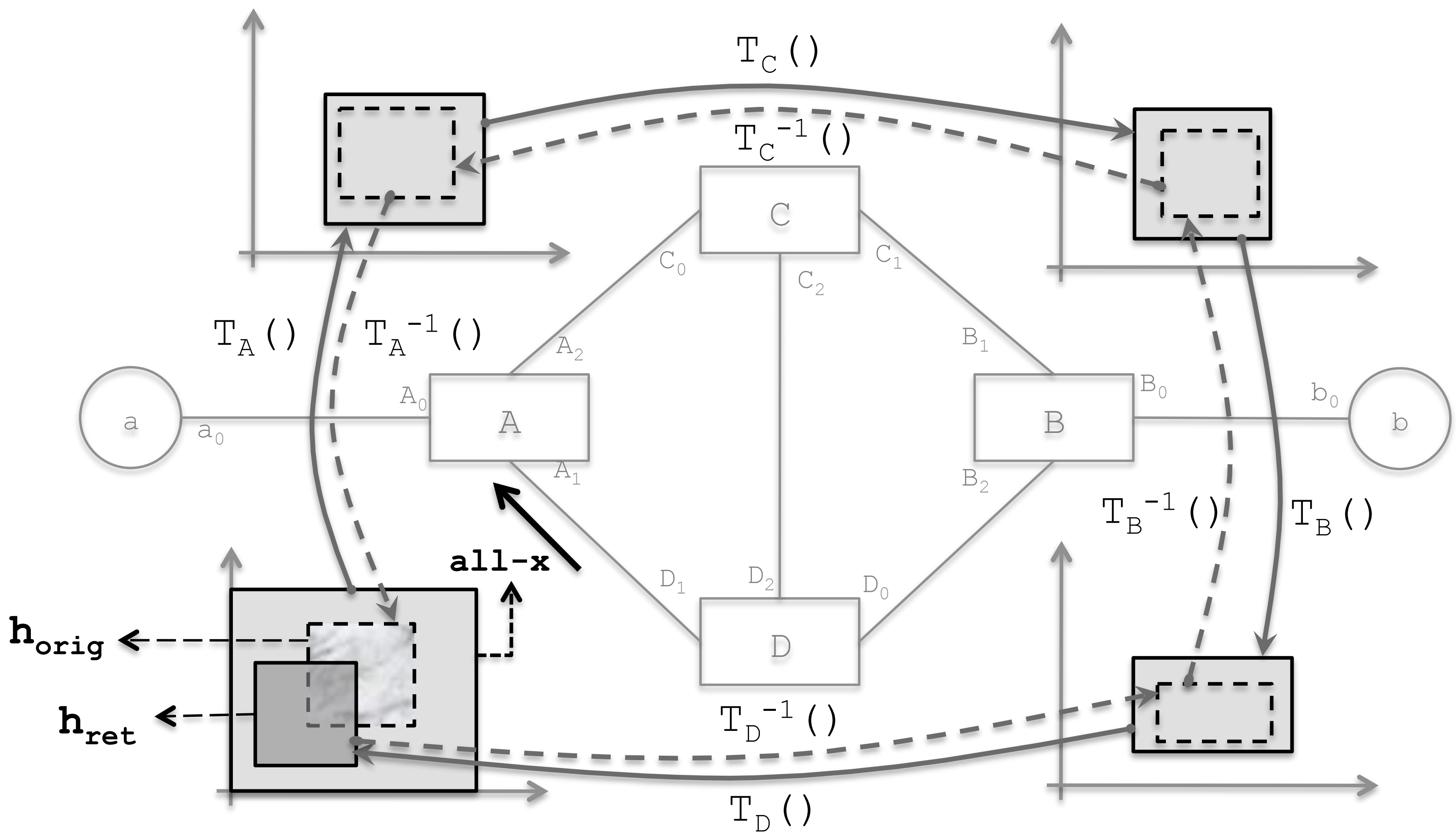
header space analysis — loop



finite loop

- $h_{ret} \cap h_{orig} = \varnothing$

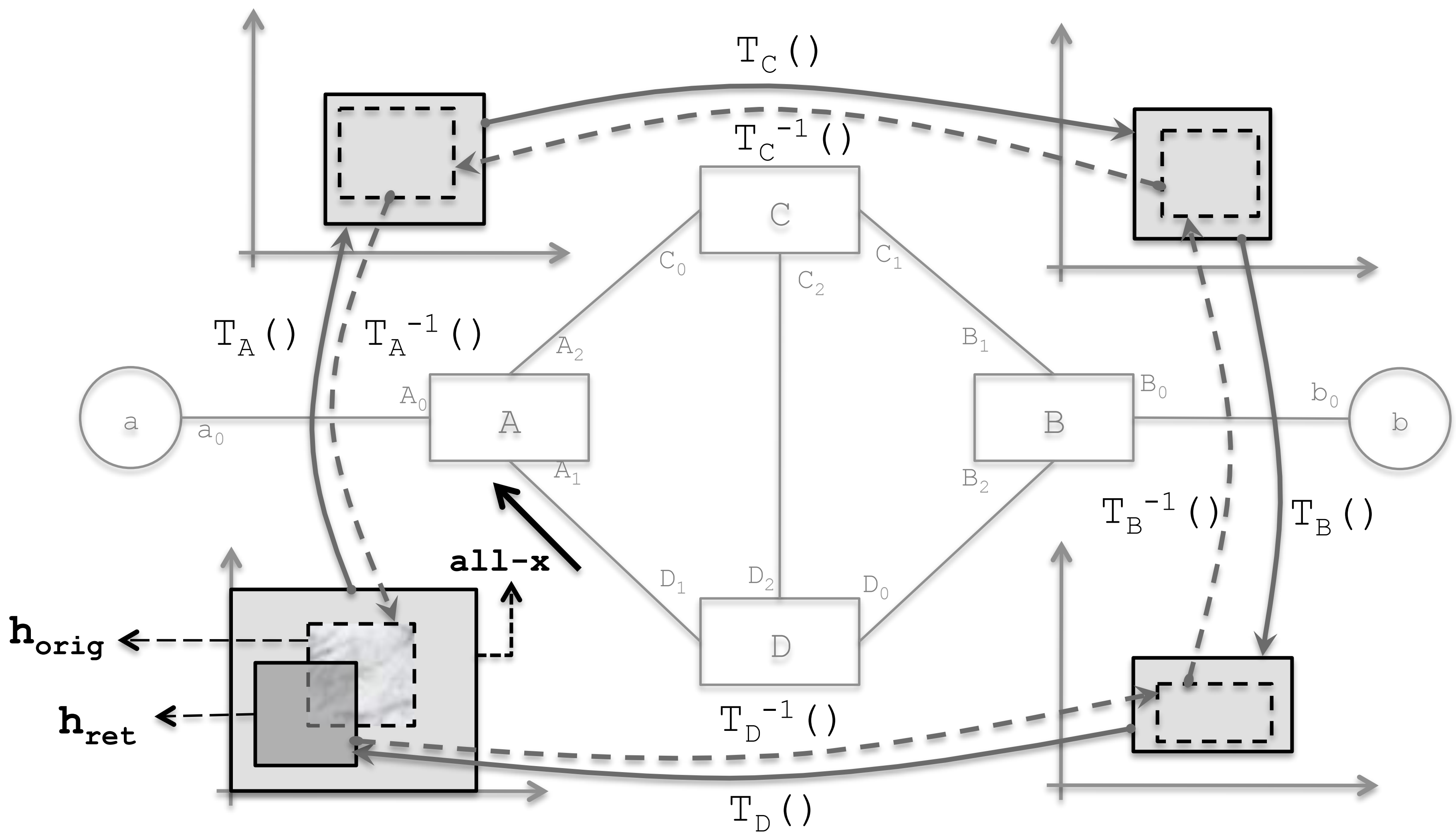
header space analysis — loop



infinite loop

- $h_{ret} \subseteq h_{orig}$

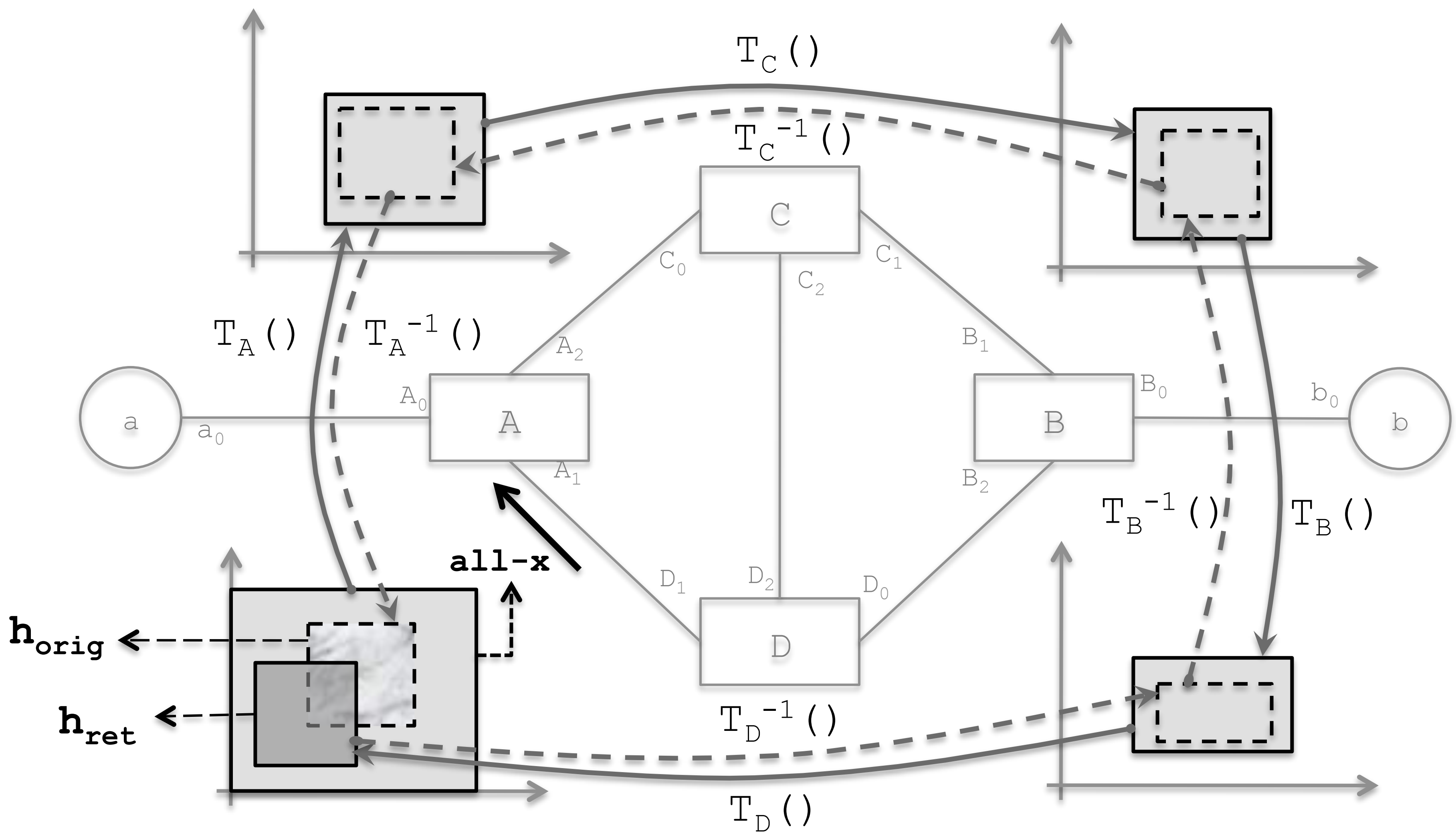
header space analysis — loop



mixed (finite and infinite)

- neither $(h_{ret} \subseteq h_{orig})$ or $h_{ret} \cap h_{orig} = \emptyset$
- h_{ret} - h_{orig} is finite loop
- examine $h_{ret} \cap h_{orig}$

header space analysis — loop



examine $h_{ret} \cap h_{orig}$

- redefine $h_{ret} := h_{ret} \cap h_{orig}$, repeat until either finite or infinite
- at most 2^L iterations

header space analysis — slice isolation

two slices, a and b with regions N_a, N_b

$$N_a = \{(\alpha_i, p_i)]_{p_i \in \mathcal{S}}\} \quad , \quad N_b = \{(\beta_i, p_i)]_{p_i \in \mathcal{S}}\}$$

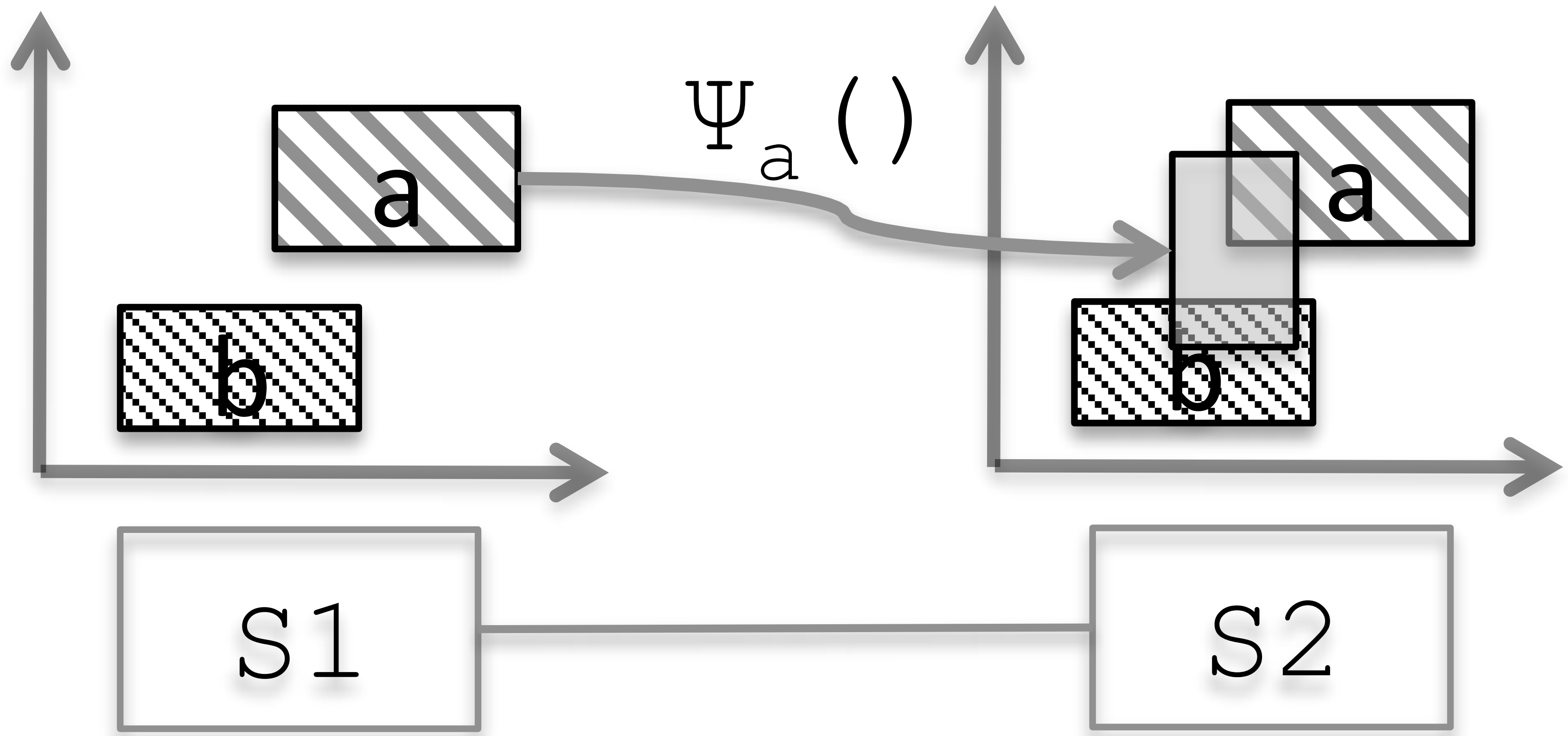
isolated $\alpha_i \cap \beta_i = \phi$

intersection

$$N_a \cap N_b = \{(\alpha_i \cap \beta_i, p_i)]_{p_i \in N_a \& p_i \in N_b}\}$$

header space analysis — slice isolation

detecting leakage



discussion

HSA is really just

- simulation + equivalence class optimization

header space analysis — loop

can packets from host a reach host b

$$R_{a \to b} = \bigcup_{a \to b \text{ paths}} \{T_n(\Gamma(T_{n-1}(......(\Gamma(T_1(h, p)...))\}$$