Implementação e avaliação da cifra de fluxo Forro14 em hardware programável Tofino usando a linguagem P4

Rodrigo Pierini, Caio Teixeira, Christian Rothenberg, Marco Henriques

SBSeg24
SÃO JOSÉ DOS CAMPOS

Realização

UNICAMP

ReGrAS
SECURITY RULES.
UNICAMP

Feec
Unicamp

INTRIG
INFORMATION & NETWORKING TECHNOLOGIES RESEARCH & INNOVATION GROUP

Apoio

SMARTNESS

ERICSSON

FAPESP

Motivação

Motivação

- Novos paradigmas de redes de computadores

Motivação

- Novos paradigmas de redes de computadores
  - *Software-Defined Networking* (SDN);

Motivação

- Novos paradigmas de redes de computadores
  - *Software-Defined Networking* (SDN);
  - *Programmable Dataplane* (PDP);

Motivação

- Novos paradigmas de redes de computadores
  - *Software-Defined Networking* (SDN);
  - *Programmable Dataplane* (PDP);
  - *In-Network Computing* (INC);

Motivação

A rede deixa de ser mera transmissora!

- Novos paradigmas de redes de computadores
  - *Software-Defined Networking* (SDN);
  - *Programmable Dataplane* (PDP);
  - *In-Network Computing* (INC);

Motivação

A rede deixa de ser mera transmissora!

- Novos paradigmas de redes de computadores
  - *Software-Defined Networking* (SDN);
  - *Programmable Dataplane* (PDP);
  - *In-Network Computing* (INC);

Como implementar criptografia nesse contexto?

Motivação

- Algoritmos de criptografia implementados *In-Network*:

Motivação

- Algoritmos de criptografia implementados *In-Network*:
  - AES [Chen, 2020];

AES-256: ~7Gbps de vazão máxima

* Vazão máxima cifrando 128b em um Intel Tofino™

Motivação

- Algoritmos de criptografia implementados

  In-Network:

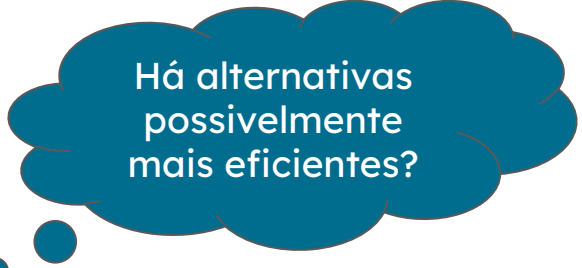
  - AES [Chen, 2020];
  - ChaCha20 [Yoshinaka *et al*, 2022].

AES-256: ~7Gbps de vazão máxima

ChaCha20: ~14Gbps de vazão máxima†

† Vazão máxima cifrando 512b em um Intel Tofino™

Motivação



- Algoritmos de criptografia implementados *In-Network*:
  - AES [Chen, 2020];
  - ChaCha20 [Yoshinaka *et al*, 2022].

Motivação

- Algoritmo de cifra de fluxo recentemente proposto: Forro14 [Coutinho *et al*, 2023]

Motivação

- Algoritmo de cifra de fluxo recentemente proposto: Forro14 [Coutinho *et al*, 2023]
  - Menos rodadas que o ChaCha20 com o mesmo nível de segurança;

Motivação

- Algoritmo de cifra de fluxo recentemente proposto: Forro14 [Coutinho *et al*, 2023]
  - Menos rodadas que o ChaCha20 com o mesmo nível de segurança;
  - Melhor desempenho em cenários sem paralelização de processamento.

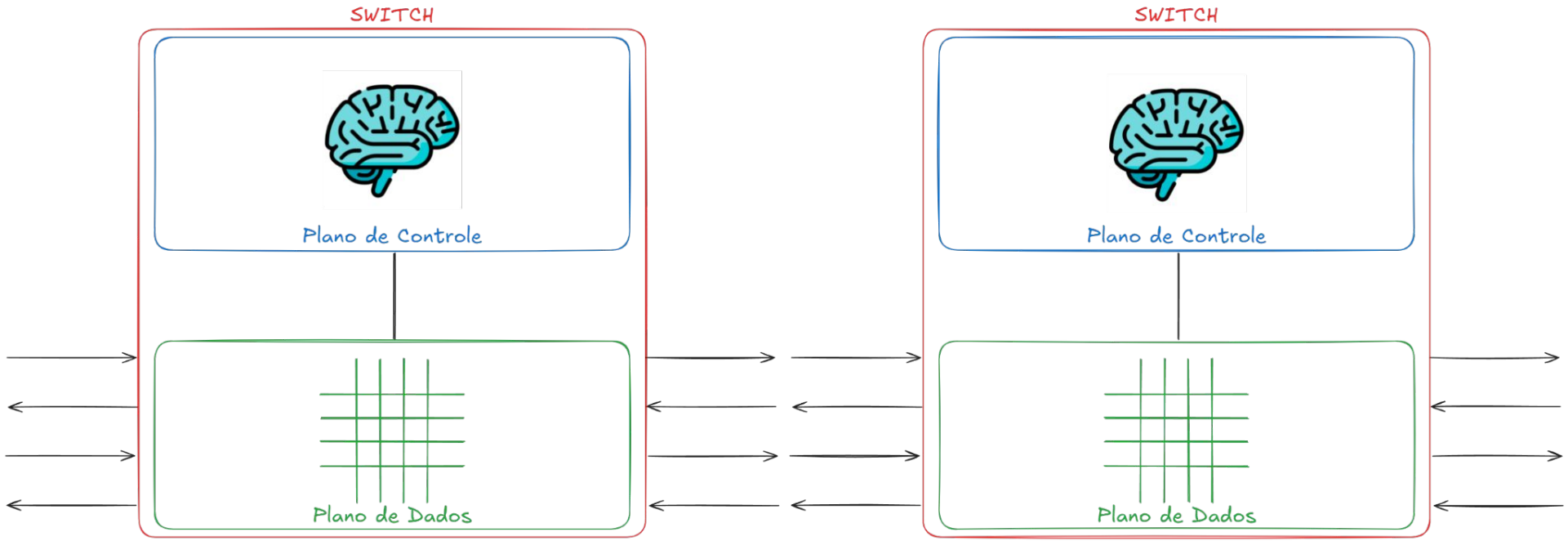
Pergunta de pesquisa

“No contexto de redes programáveis, a cifra de fluxo Forro14 é uma alternativa mais eficiente que a cifra ChaCha20 em uso de recursos e em vazão de dados?”

Fundamentos: Princípios de Software-Defined Networking

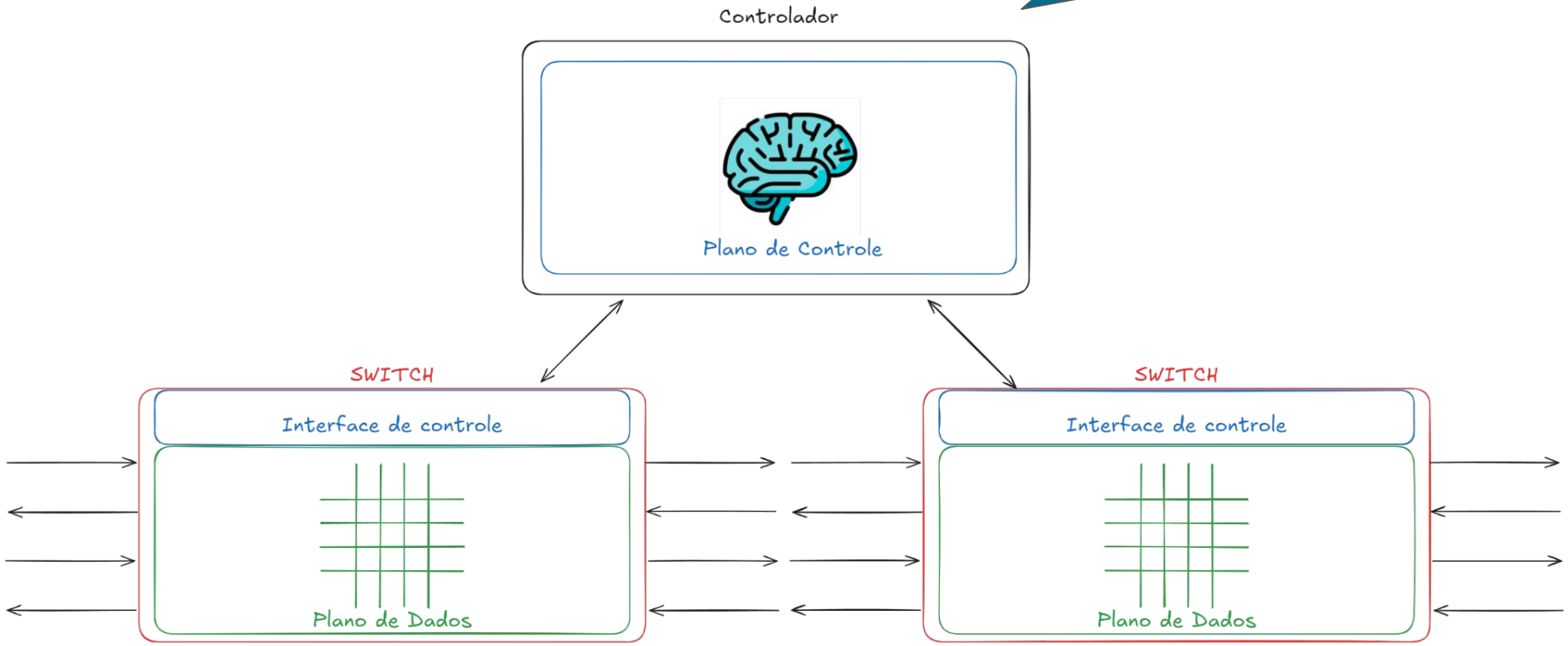
Fundamentos: *SDN*

Rede tradicional



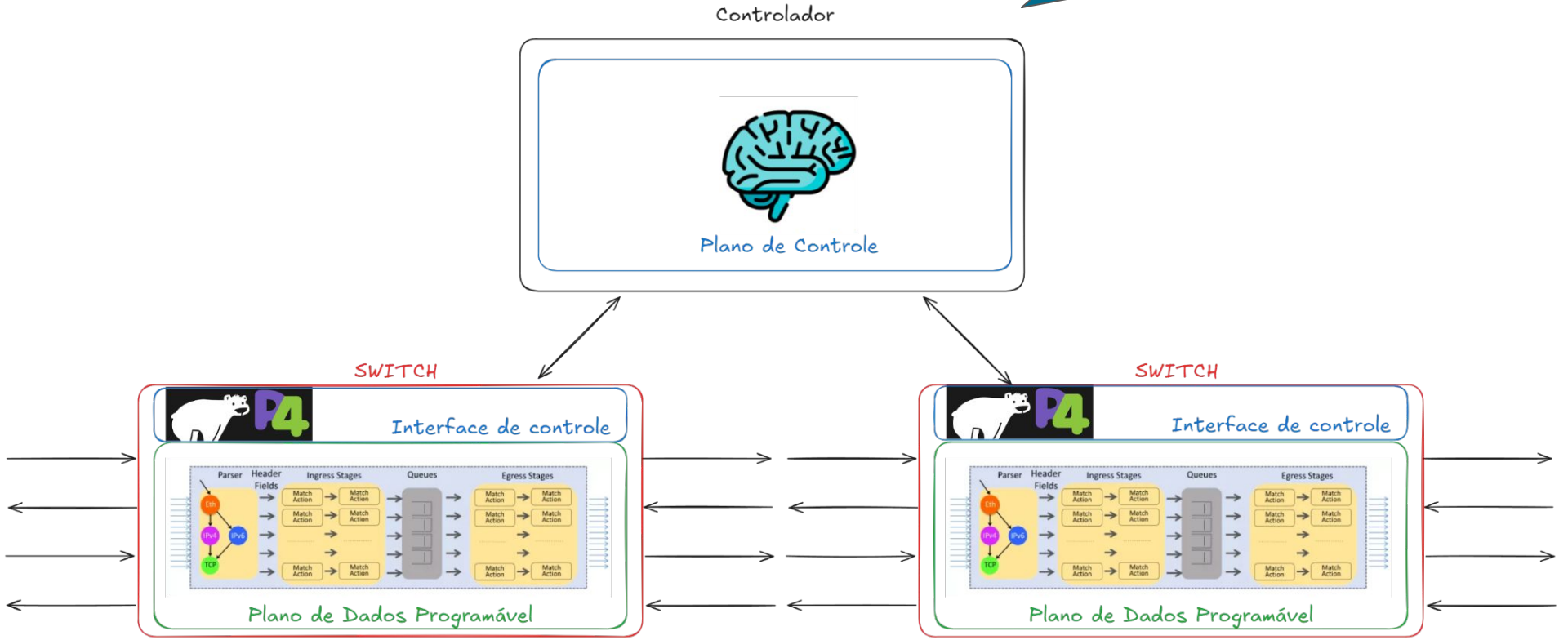
Fundamentos: *SDN*

Redes Definidas por Software (SDN)



Fundamentos: *PDP*

Plano de Dados Programável (PDP)

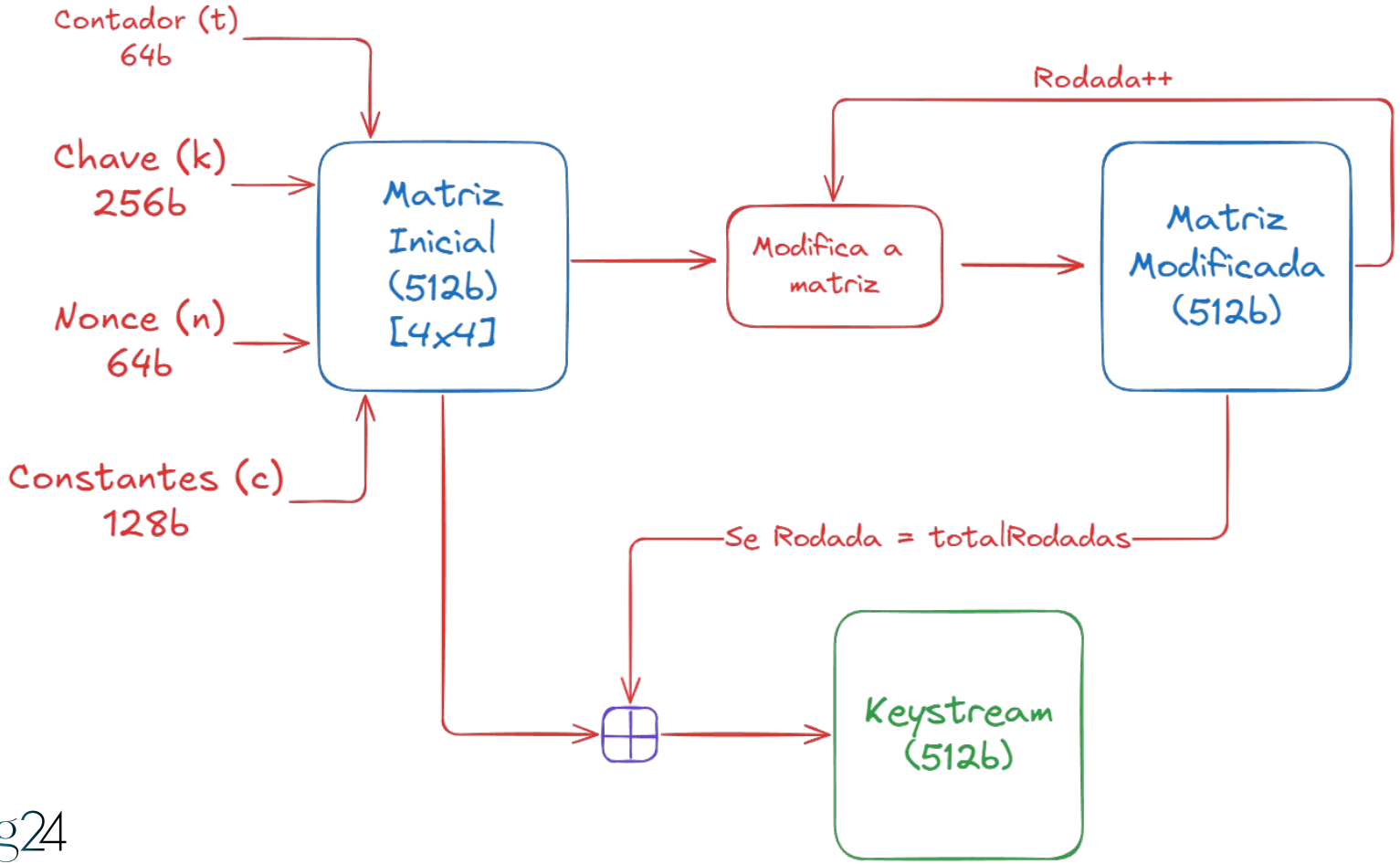


Fonte da imagem: https://www.systemsapproach.org/blog-archive/sdn-and-disaggregation

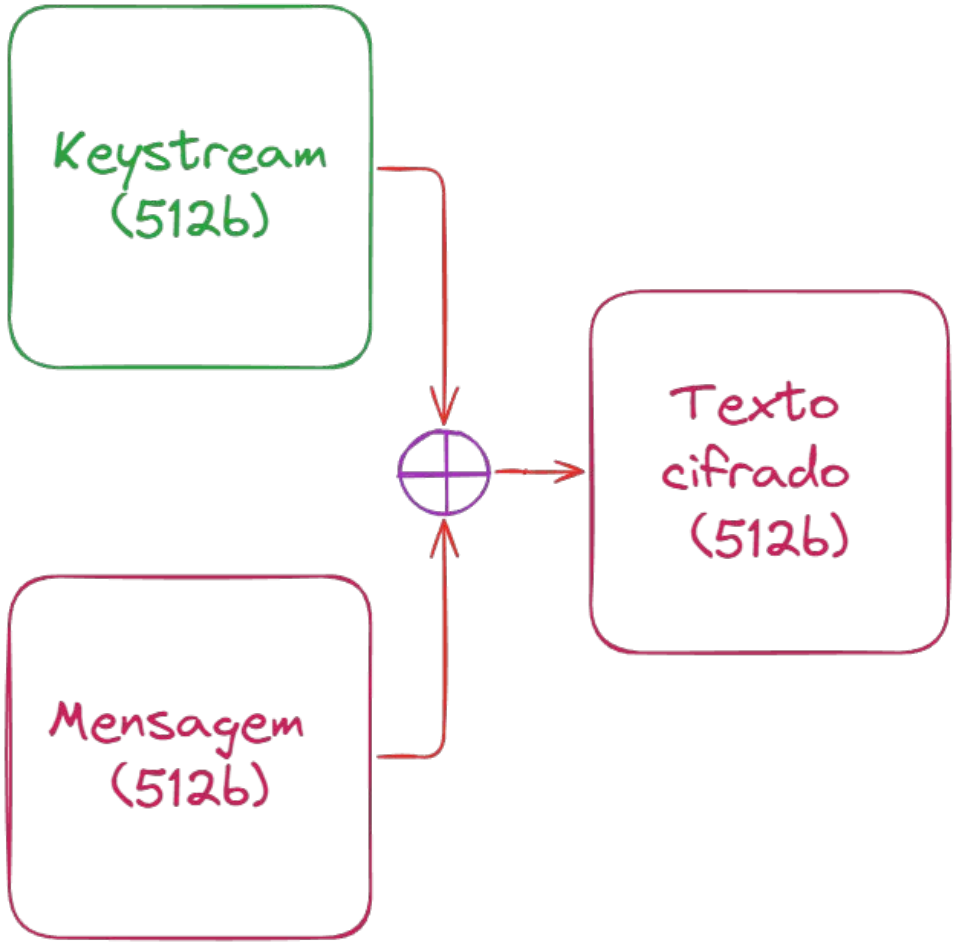
Fundamentos
Cifras de Fluxo

Focado no ChaCha20 e Forro14

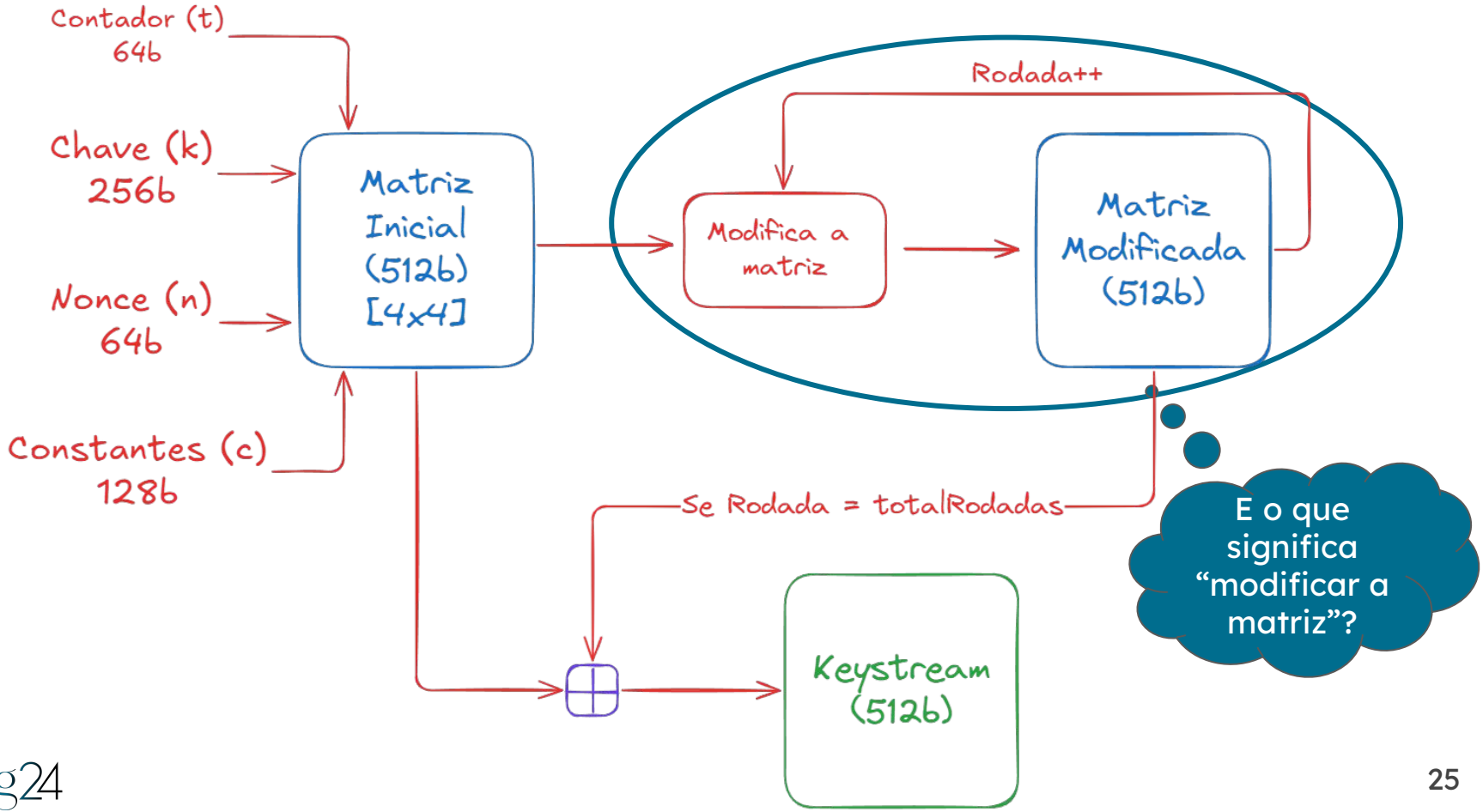
Fundamentos: *ChaCha20 e Forro14*



Fundamentos: *ChaCha20 e Forro14*

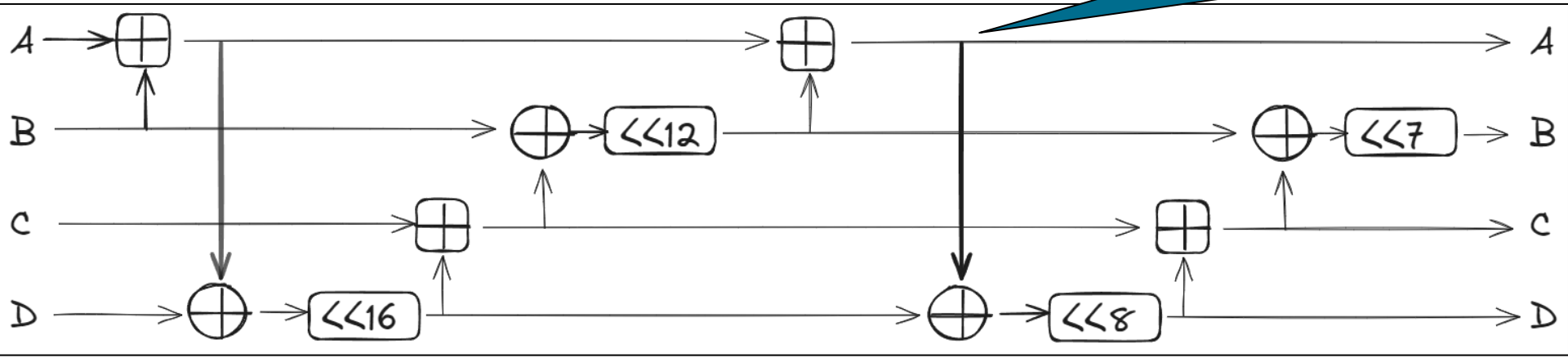


Fundamentos: *ChaCha20 e Forro14*



Fundamentos: *ChaCha20*

Quarter-Round Function (QR)



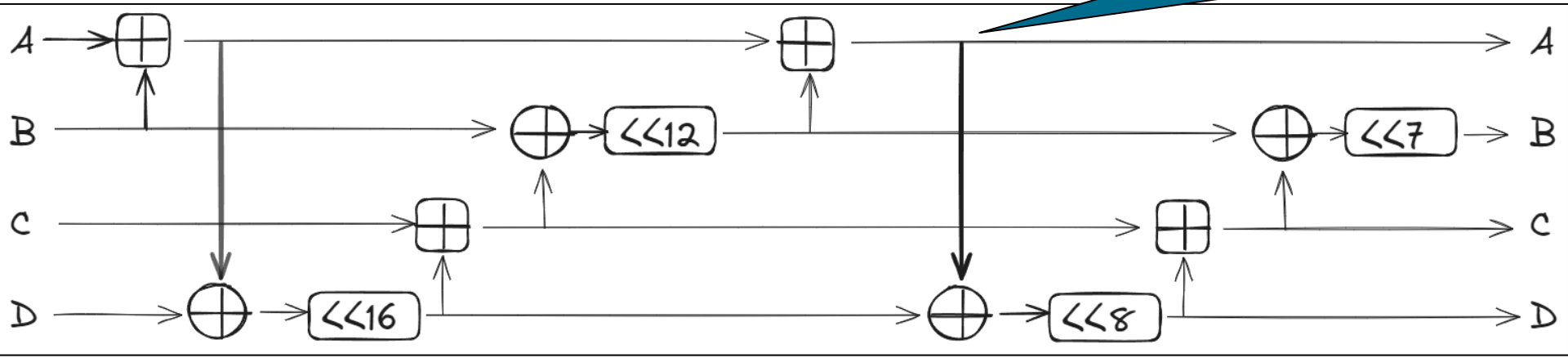
adição mod 2^32

bitwise XOR

<<<N Rotação circular à esquerda

Fundamentos: *ChaCha20*

Quarter-Round Function (QR)



⊞ adição mod 2^32

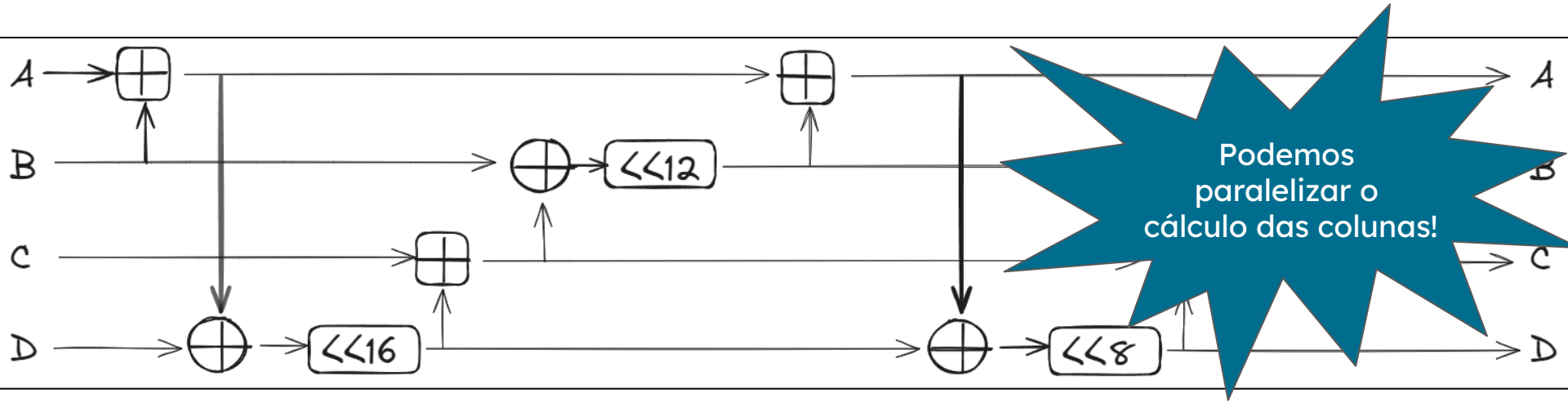
⊕ bitwise XOR

<<<N Rotação circular à esquerda

	0	1	2	3
A	c0	c1	c2	c3
B	k0	k1	k2	k3
C	k4	k5	k6	k7
D	t0	t1	n0	n1

c = constante
k = chave
t = contador
n = nonce

Fundamentos: *ChaCha20*



⊞ adição mod 2^32

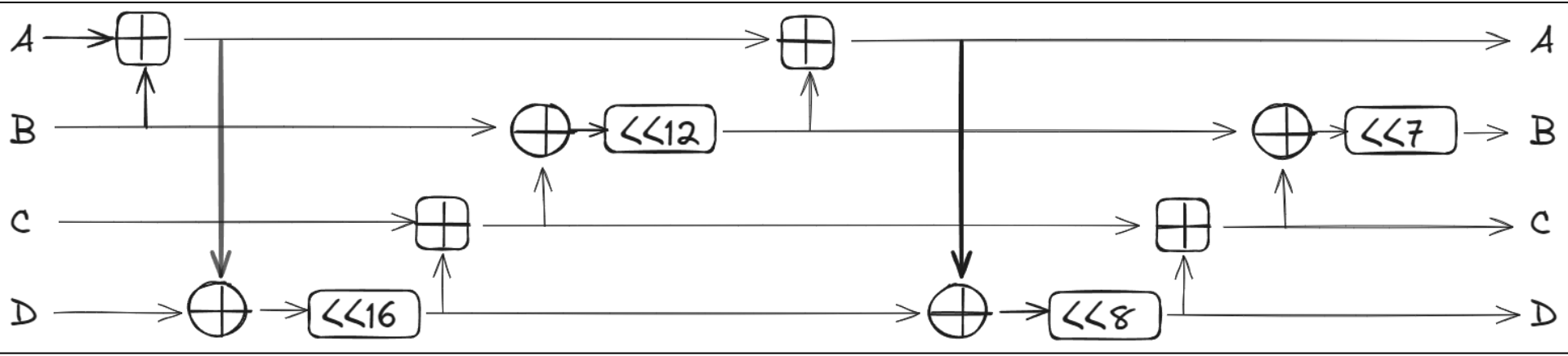
⊕ bitwise XOR

<<<N Rotação circular à esquerda

	0	1	2	3
A	c0	c1	c2	c3
B	k0	k1	k2	k3
C	k4	k5	k6	k7
D	t0	t1	n0	n1

c = constante
k = chave
t = contador
n = nonce

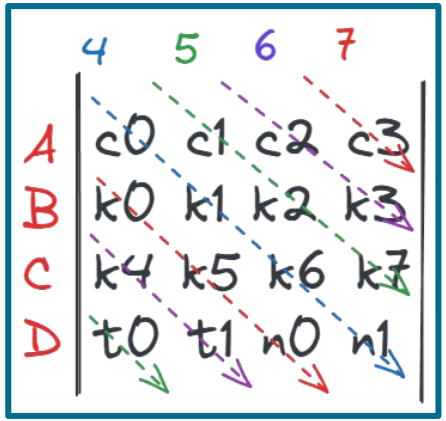
Fundamentos: *ChaCha20*



⊞ adição mod 2^32

⊕ bitwise XOR

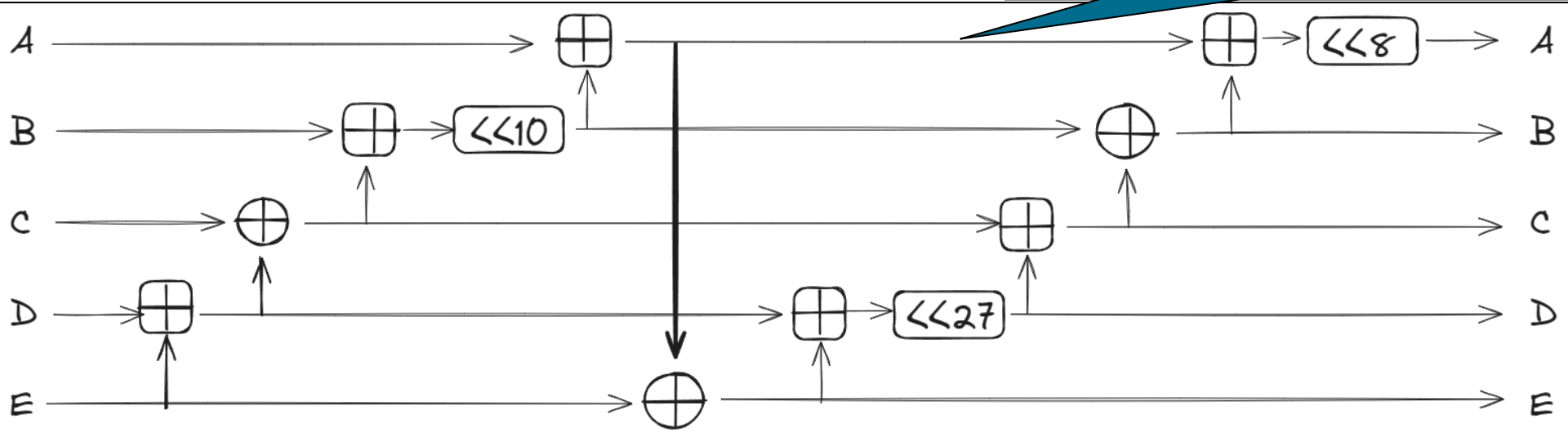
<<<N Rotação circular à esquerda



c = constante
k = chave
t = contador
n = nonce

Fundamentos: *Forro14*

Quarter-Round Function (QR)



⊞ adição mod 2^32

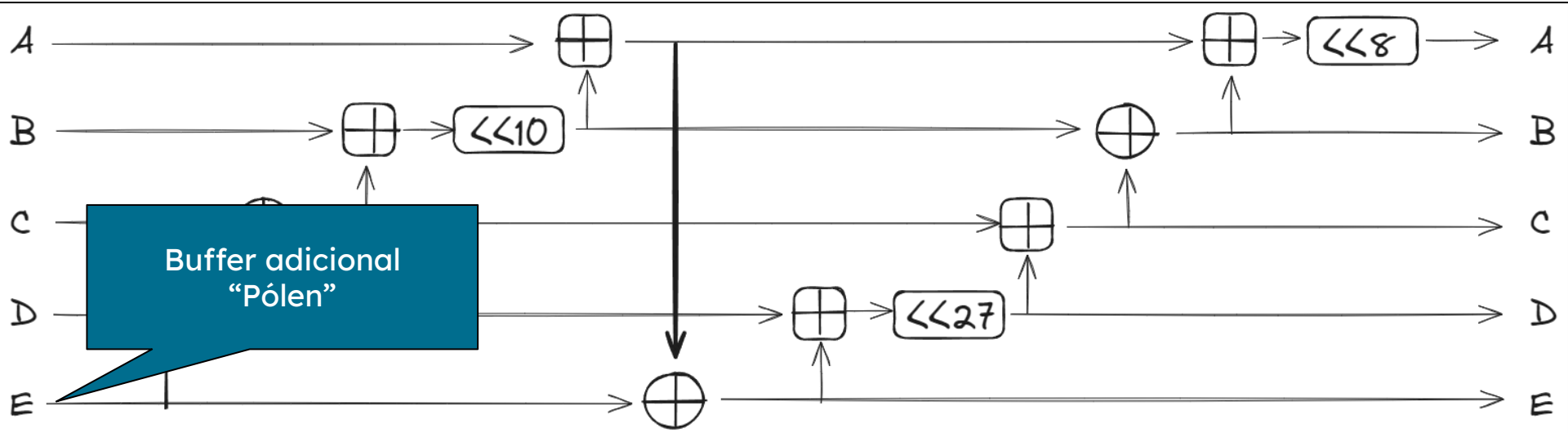
⊕ bitwise XOR

<<<N Rotação circular à esquerda

	0	1	2	3
A	k0	k1	k2	k3
B	t0	t1	c0	c1
C	k4	k5	k6	k7
D	n0	n1	c2	c3

c = constante
k = chave
t = contador
n = nonce

Fundamentos: *Forro14*



⊕ adição mod 2^32

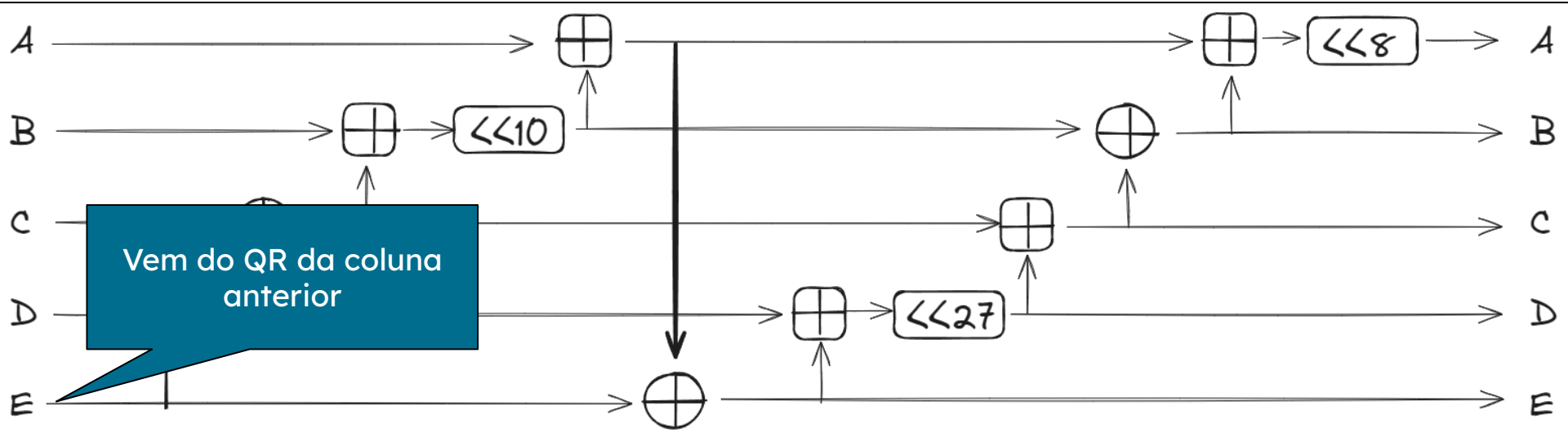
⊕ bitwise XOR

<<N Rotação circular à esquerda

	0	1	2	3	E
A	k0	k1	k2	k3	
B	t0	t1	c0	c1	
C	k4	k5	k6	k7	
D	n0	n1	c2	c3	

c = constante
k = chave
t = contador
n = nonce

Fundamentos: *Forro14*



adição mod 2^32

bitwise XOR

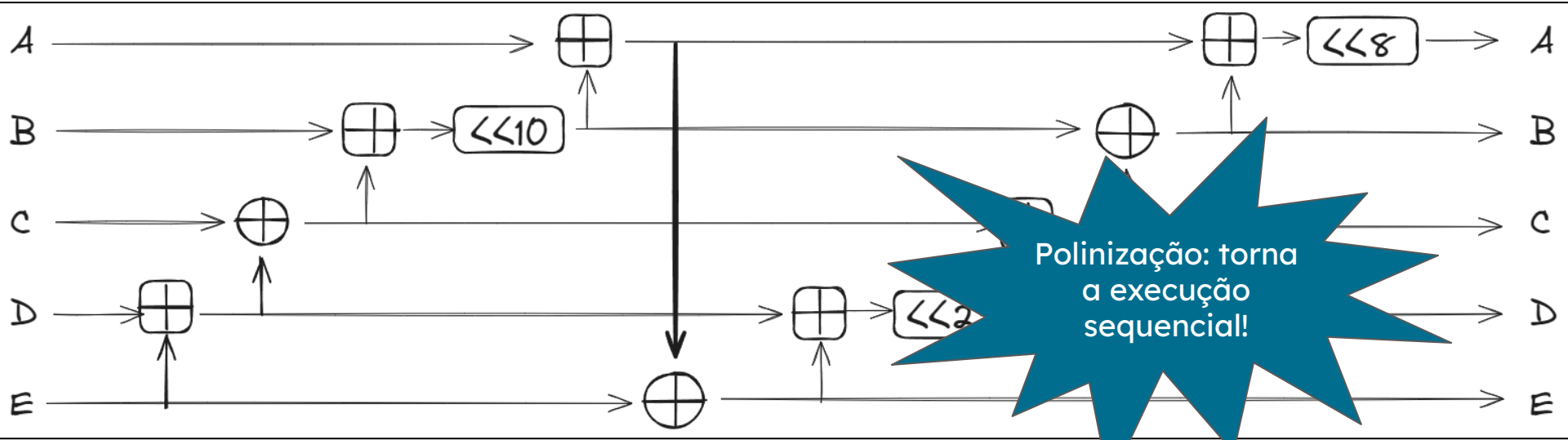
<<<N Rotação circular à esquerda

E

	0	1	2	3
A	k0	k1	k2	k3
B	t0	t1	c0	c1
C	k4	k5	k6	k7
D	n0	n1	c2	c3

c = constante
k = chave
t = contador
n = nonce

Fundamentos: *Forro14*



⊞ adição mod 2^32

⊕ bitwise XOR

<<N Rotação circular à esquerda

E

	0	1	2	3
A	k0	k1	k2	k3
B	t0	t1	c0	c1
C	k4	k5	k6	k7
D	n0	n1	c2	c3

c = constante
k = chave
t = contador
n = nonce

Desafios e solução proposta

Desafios

- Tofino1: 2 pipelines de 12 estágios;
  - QR: 12 operações;

Desafios

- Tofino1: 2 pipelines de 12 estágios;
  - QR: 12 operações;

Desafios

- Tofino1: 2 pipelines de 12 estágios;
  - QR: 12 operações;
- Alocação de memória
  - Dependências dificultam a alocação da memória no pipeline pelo compilador.

Desafios

- Tofino1: 2 pipelines de 12 estágios;
  - QR: 12 operações;
- Alocação de memória
  - Dependências dificultam a alocação da memória no pipeline pelo compilador.

A compilação pode falhar!

Desafios

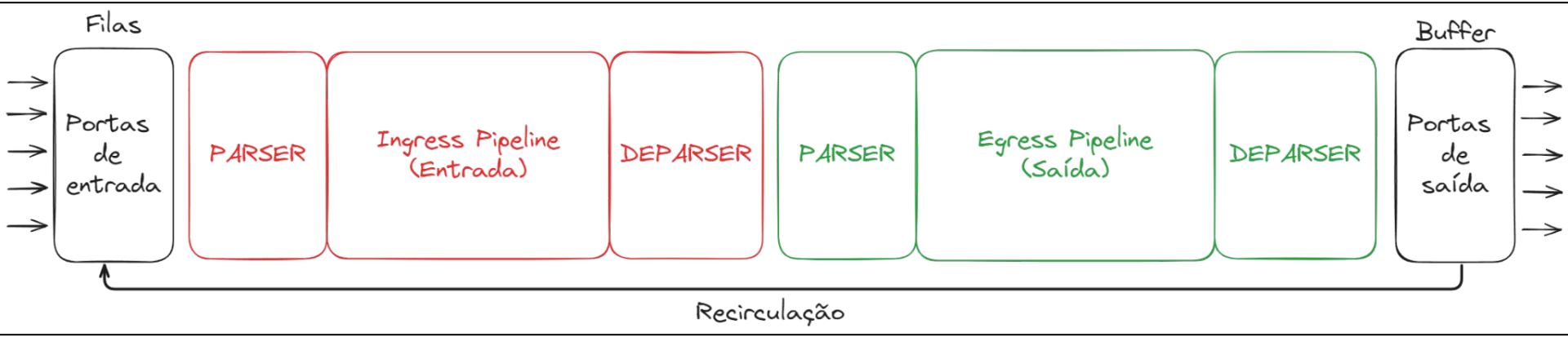
- Tofino1: 2 pipelines de 12 estágios;
  - QR: 12 operações;
- Alocação de memória
  - Dependências dificultam a alocação da memória no pipeline pelo compilador.
- Sem loops disponíveis (P4)
  - Utilizamos recirculação de pacotes.

Desafios

- Tofino1: 2 pipelines de 12 estágios;
  - QR: 12 operações;
- Alocação de memória
  - Dependências dificultam a alocação da memória no pipeline pelo compilador.
- Sem loops disponíveis (P4)
  - Utilizamos recirculação de pacotes.

Impactos na vazão!

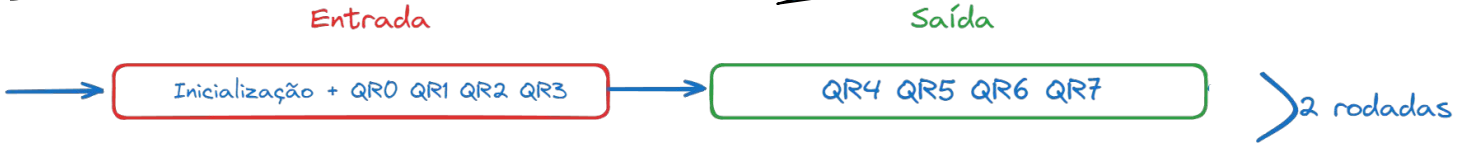
Desafios



Arquitetura referência de um switch programável com P4

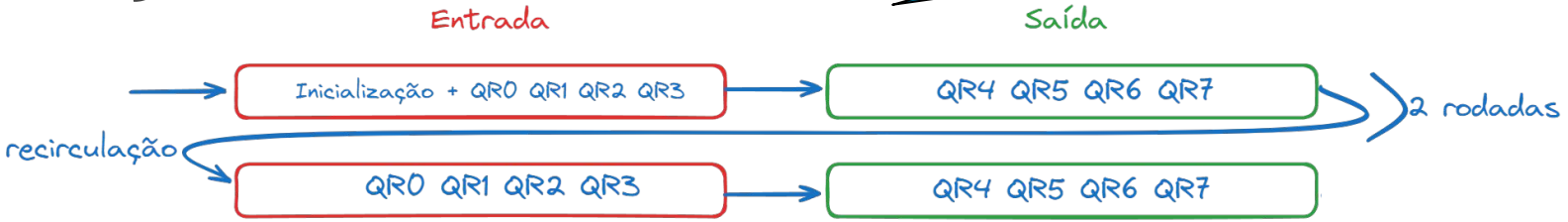
Solução Proposta

Implementação do Algoritmo ChaCha20 paralelizado



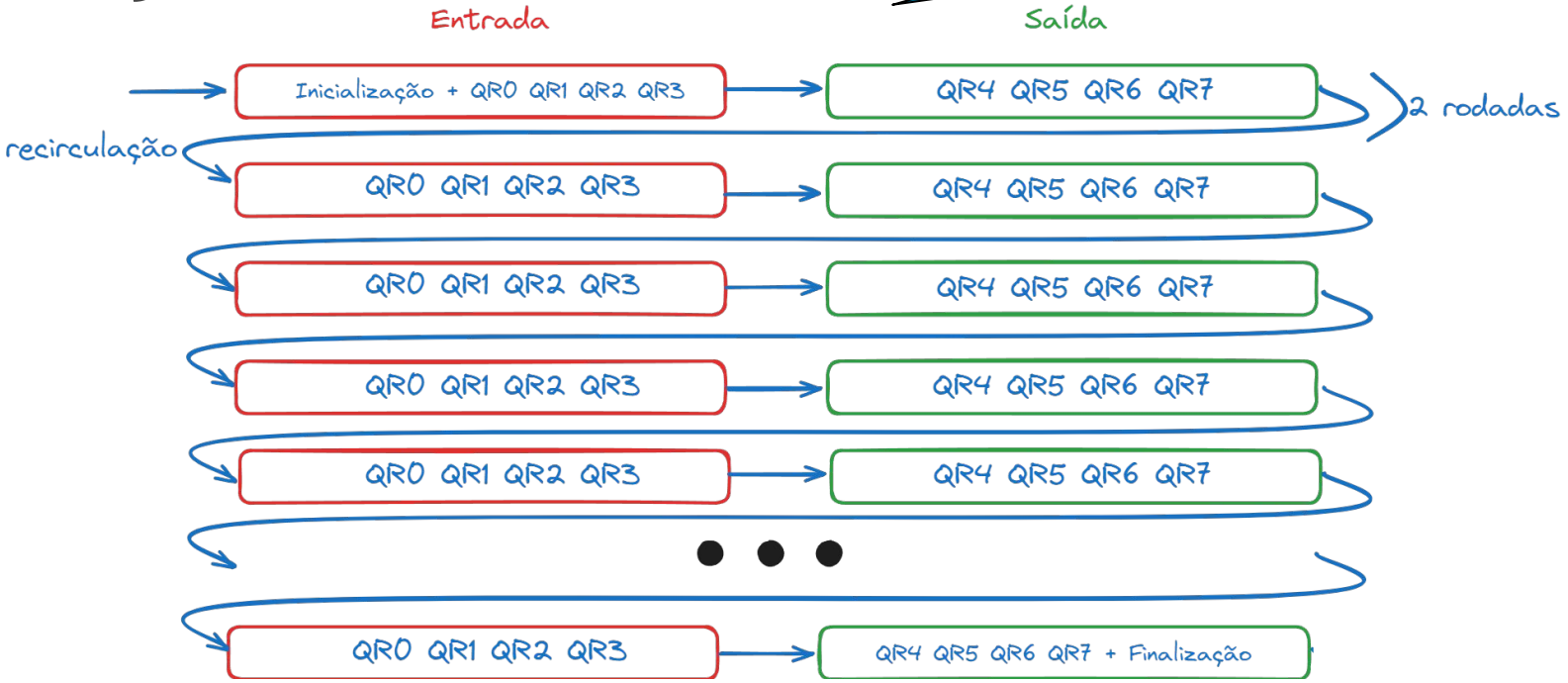
Solução Proposta

Implementação do Algoritmo ChaCha20 paralelizado



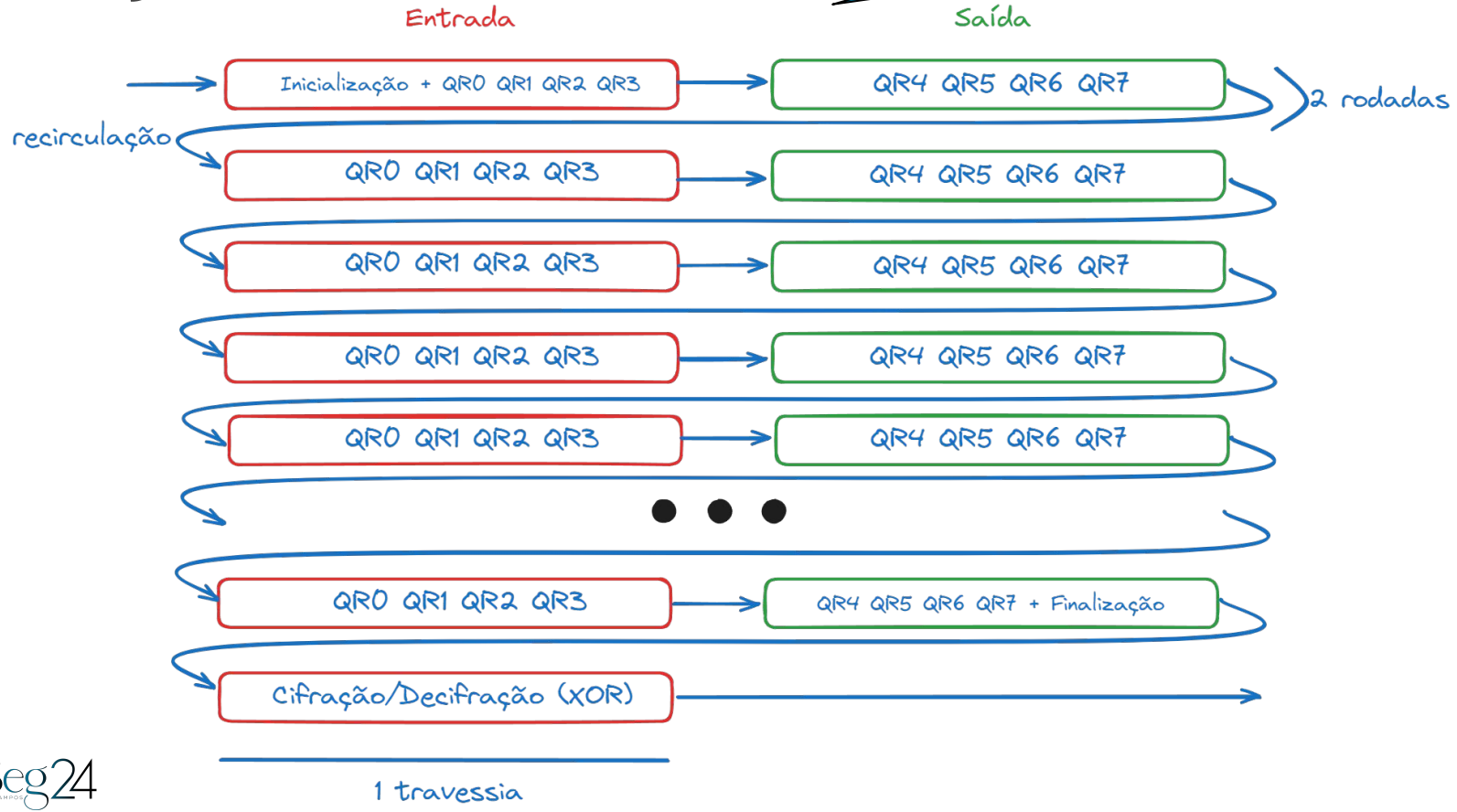
Solução Proposta

Implementação do Algoritmo ChaCha20 paralelizado



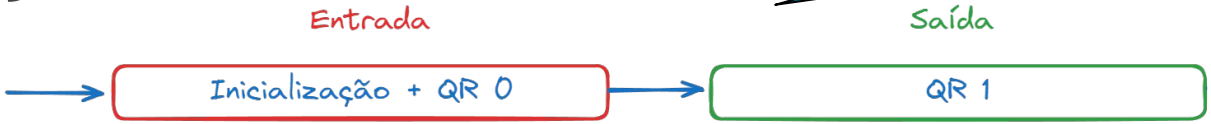
Solução Proposta

Implementação do Algoritmo ChaCha20 paralelizado



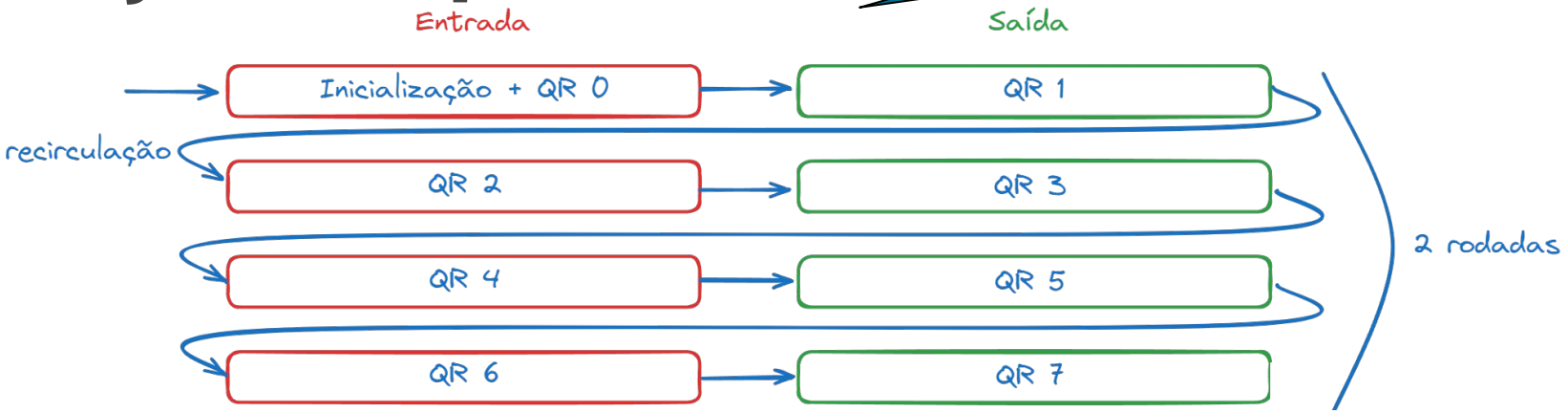
Solução Proposta

Implementação do Algoritmo Forro14



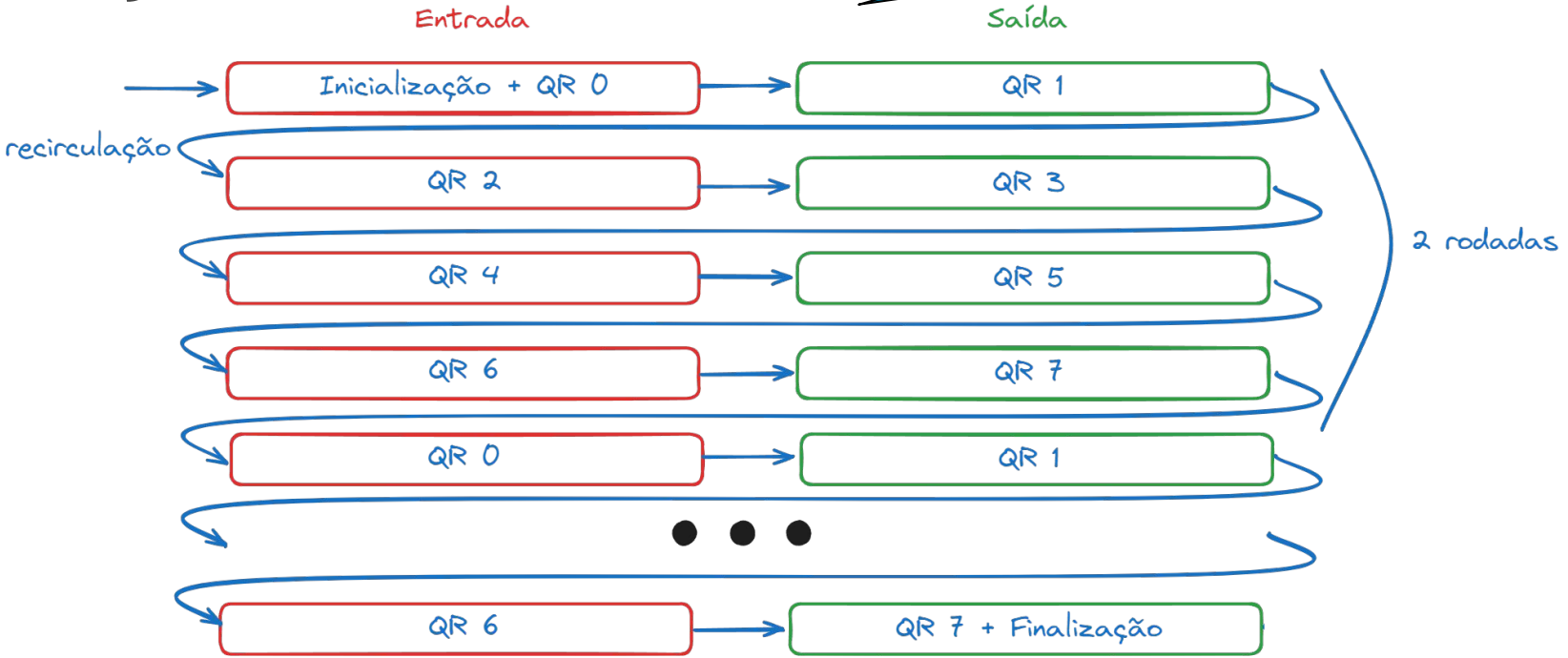
Solução Proposta

Implementação do Algoritmo Forro14



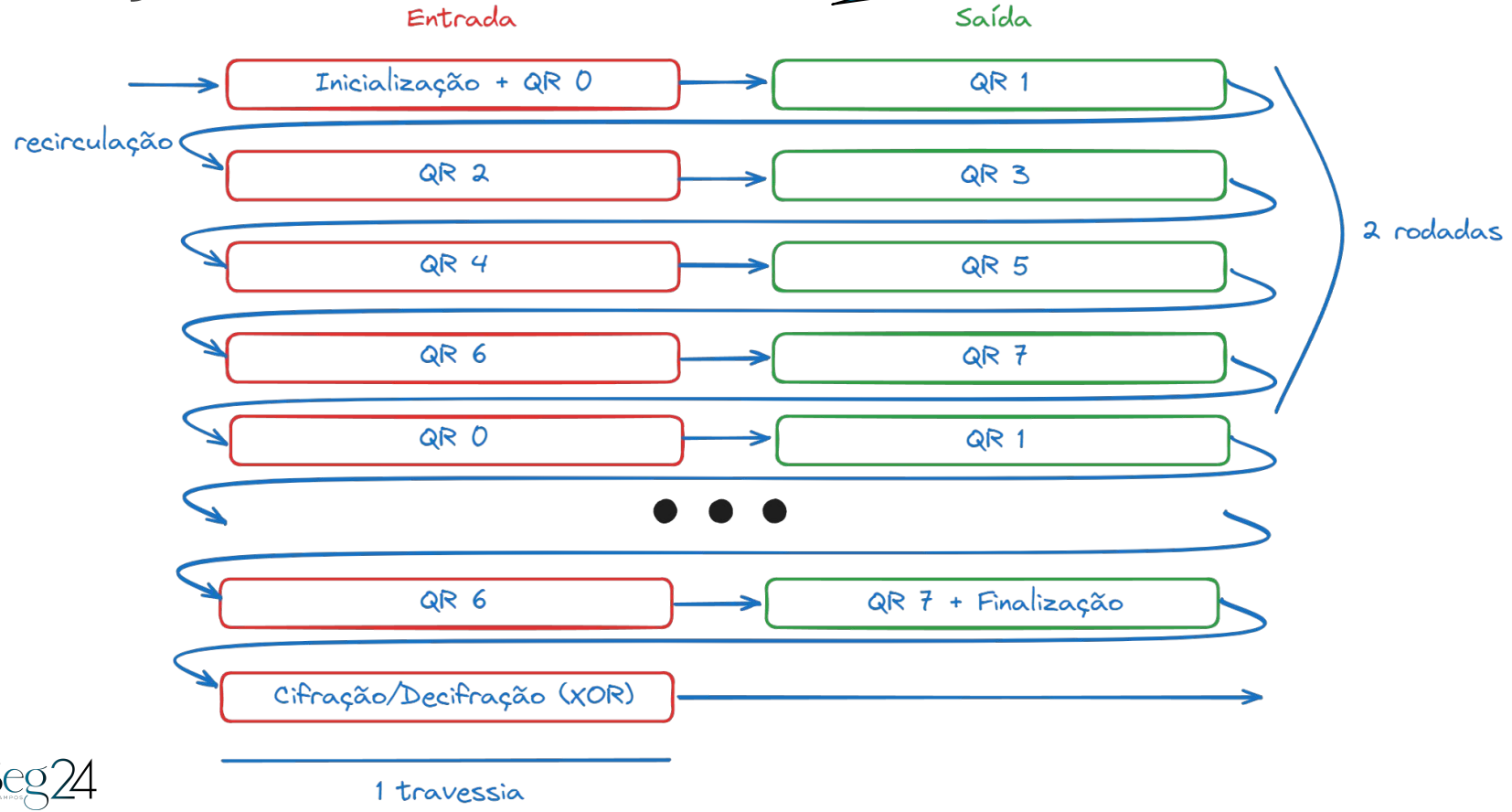
Solução Proposta

Implementação do Algoritmo Forro14



Solução Proposta

Implementação do Algoritmo Forro14



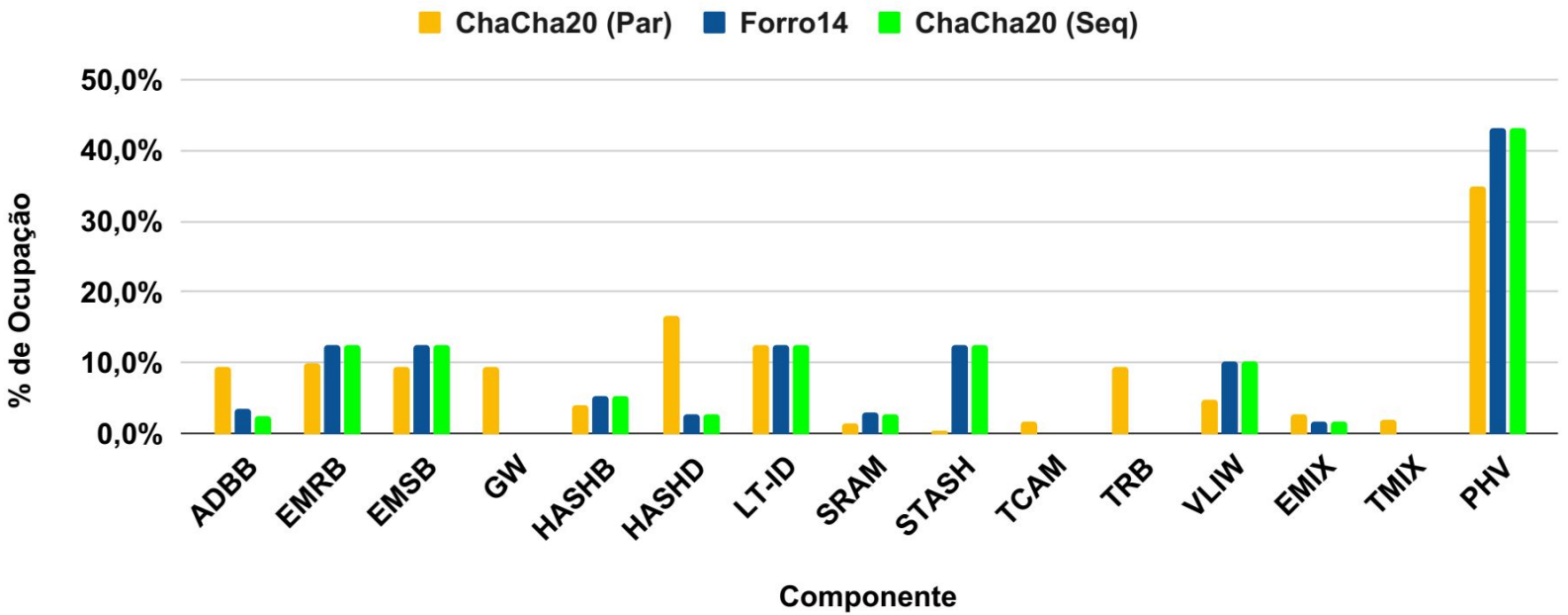
Avaliação de desempenho

Uso de recursos e vazão

Avaliação: *Implementações*

- ChaCha20 com paralelização [Yoshinaka *et al*, 2022]
  - Quatro QR por travessia do pipeline
- Forro14 [este trabalho]
  - Um QR por travessia
- ChaCha20 sequencial [este trabalho]
  - Um QR por travessia

Avaliação: *Uso de Recursos*

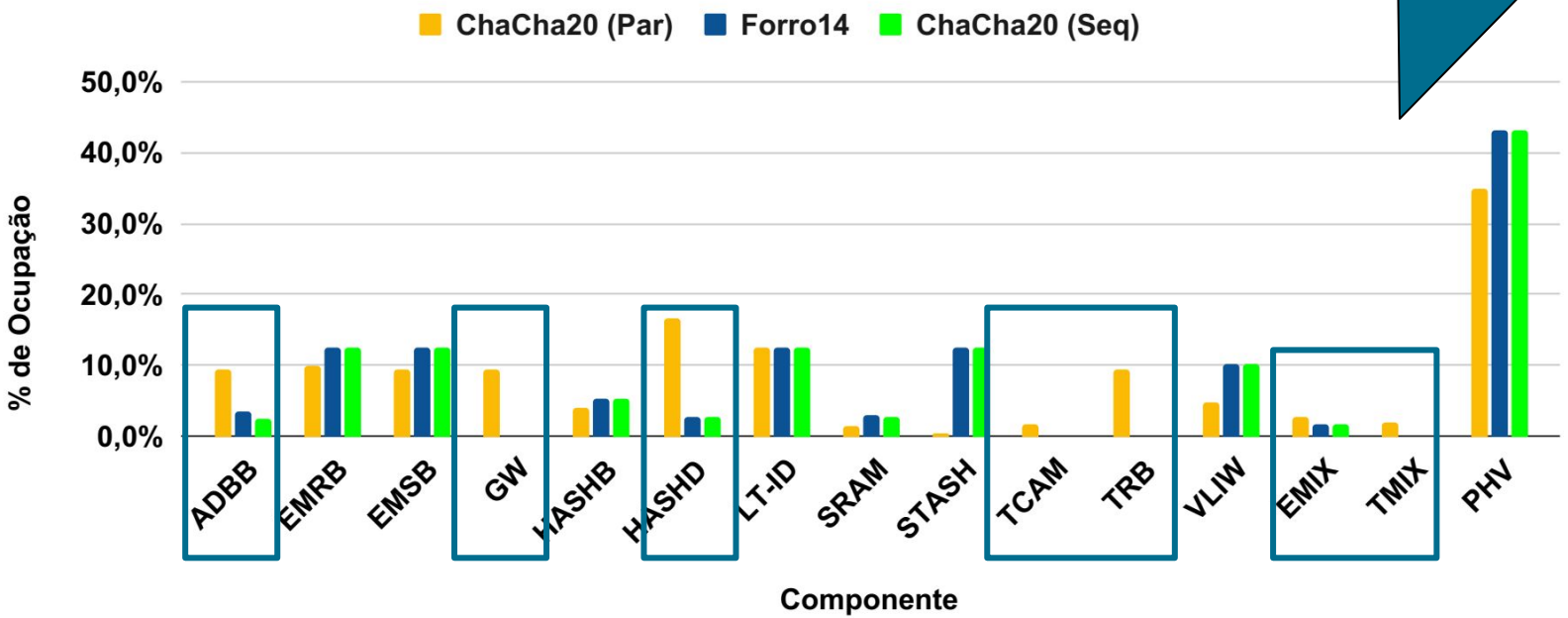


Uso dos recursos do Switch

(quanto menor, melhor)

Avaliação: *Uso de Recur*

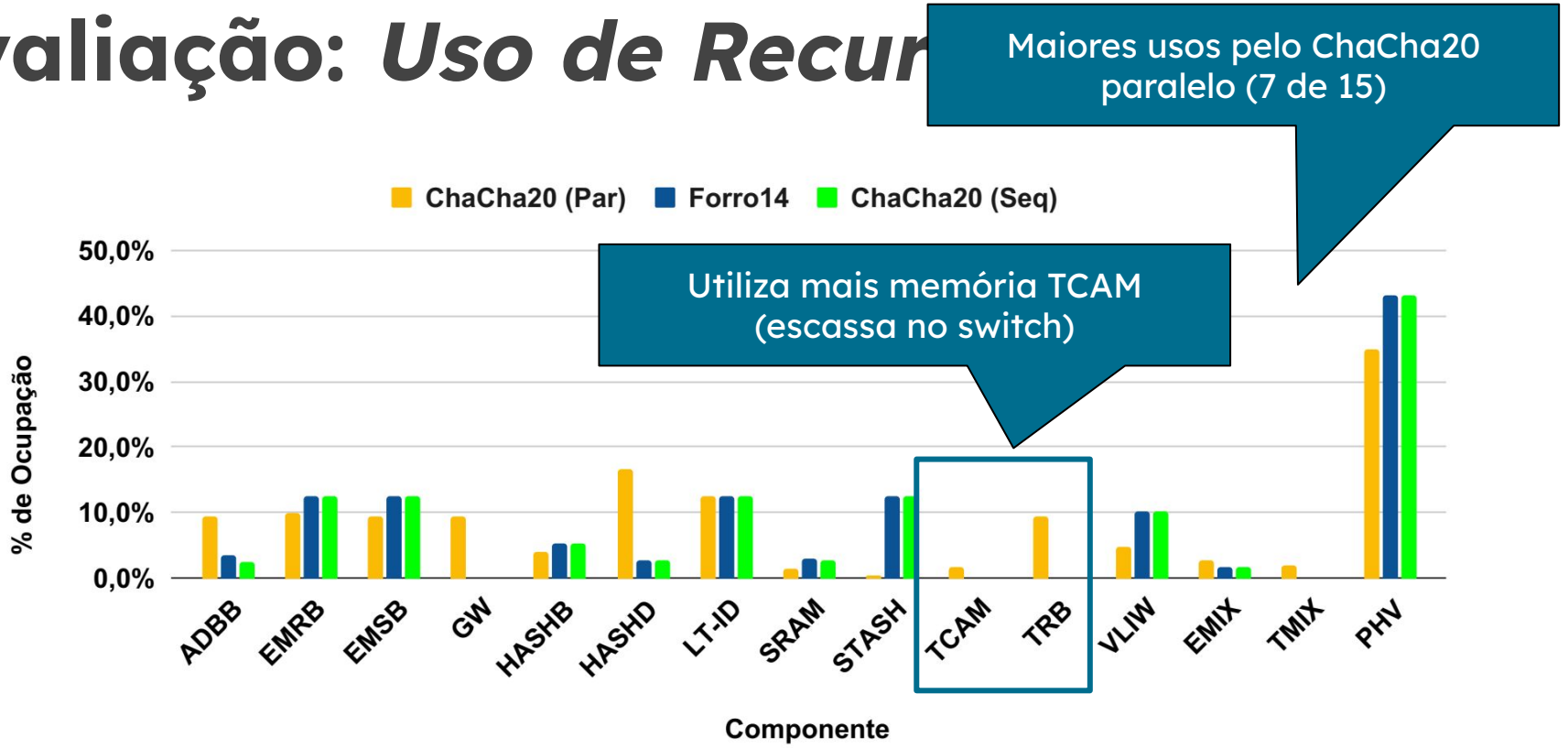
Maiores usos pelo ChaCha20 paralelo (7 de 15)



Uso dos recursos do Switch

(quanto menor, melhor)

Avaliação: *Uso de Recur*

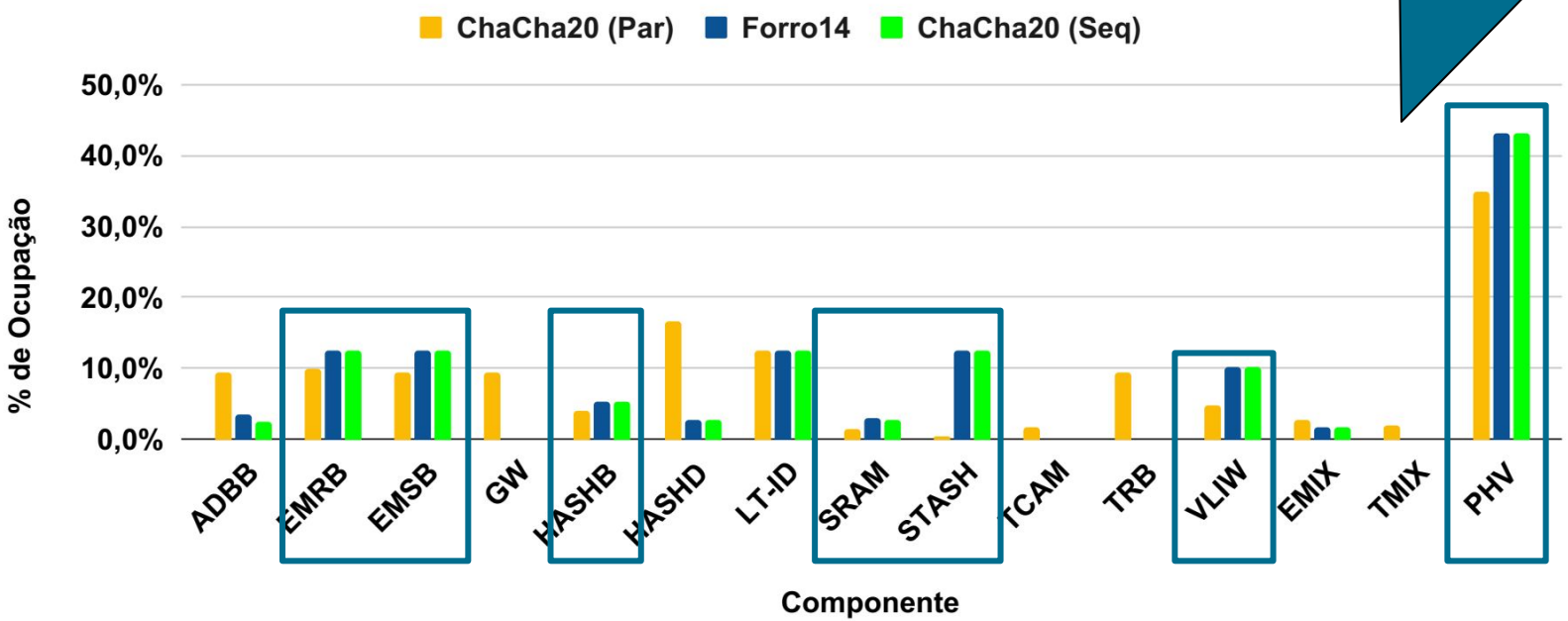


Uso dos recursos do Switch

(quanto menor, melhor)

Avaliação: *Uso de Recur*

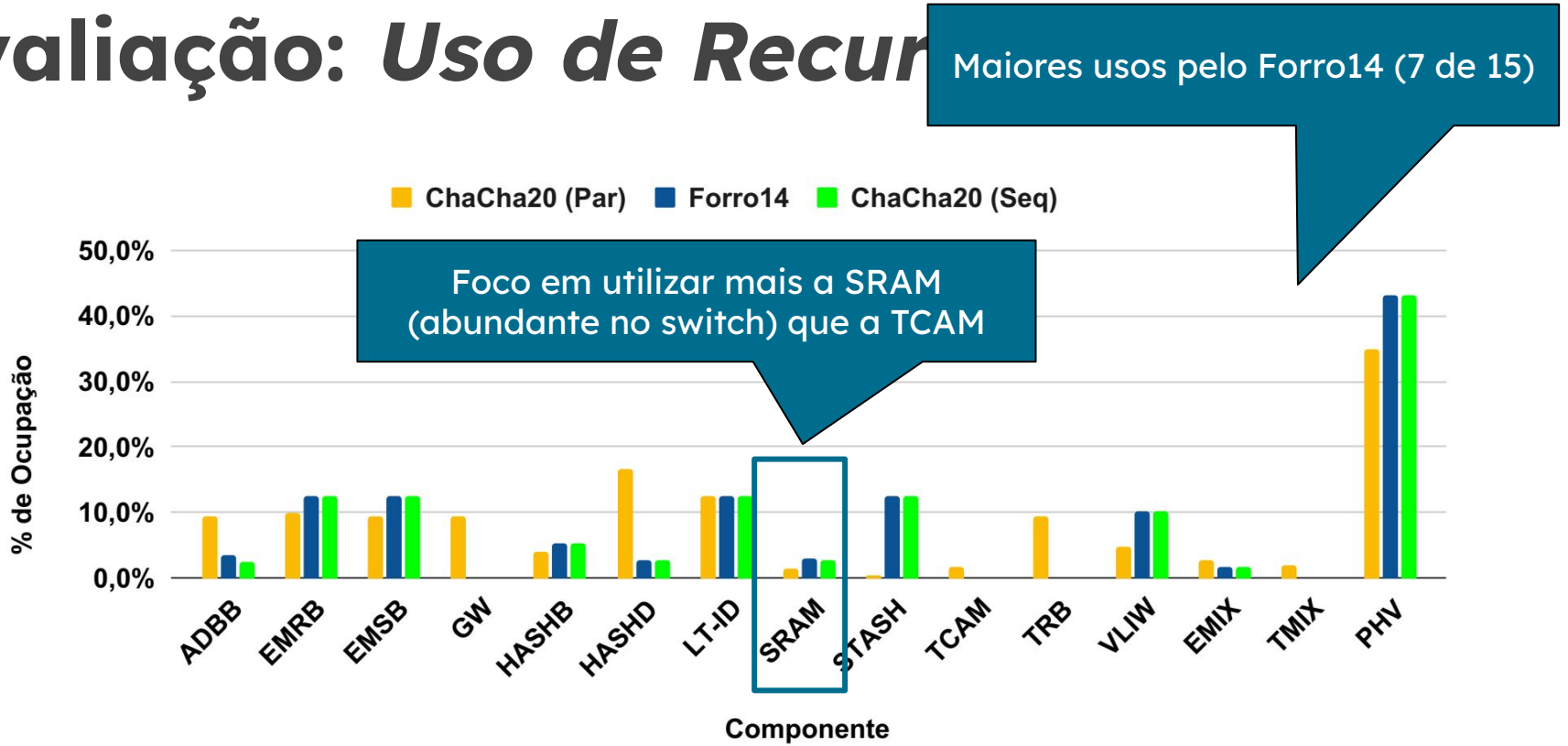
Maiores usos pelo Forro14 (7 de 15)



Uso dos recursos do Switch

(quanto menor, melhor)

Avaliação: *Uso de Recur*

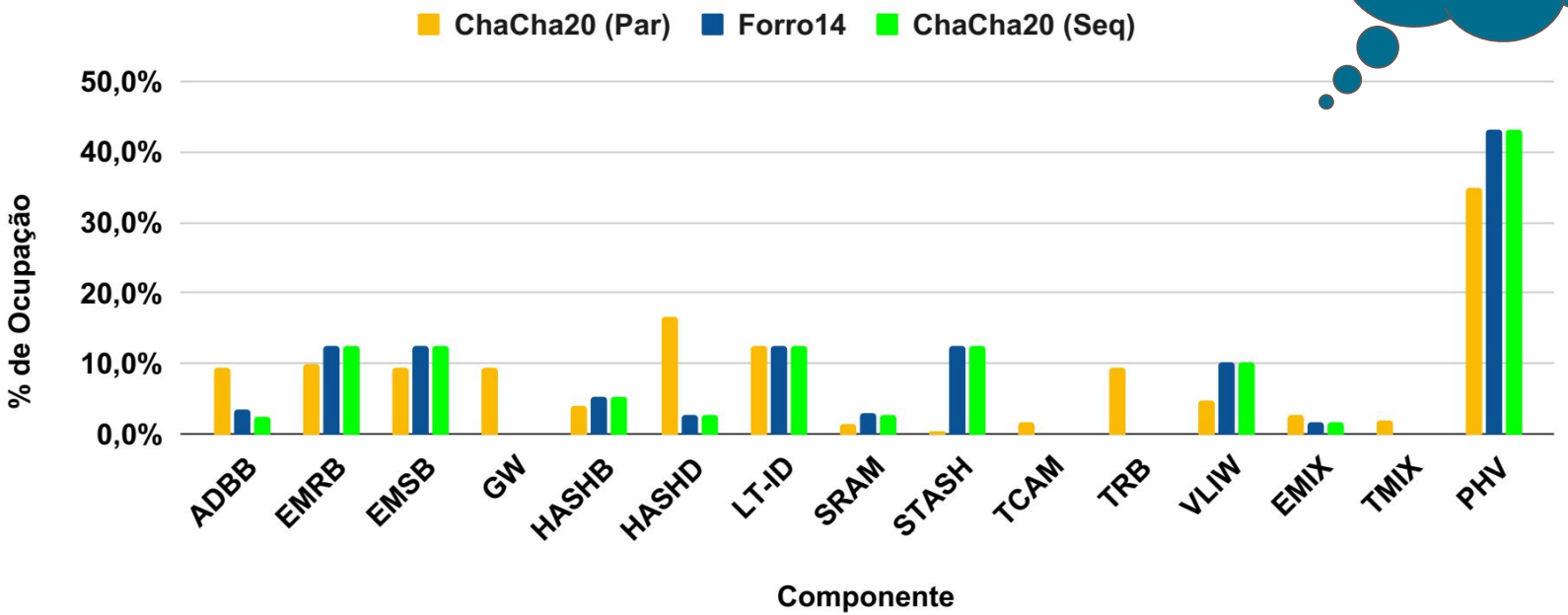


Uso dos recursos do Switch

(quanto menor, melhor)

Avaliação: *Uso de Recursos*

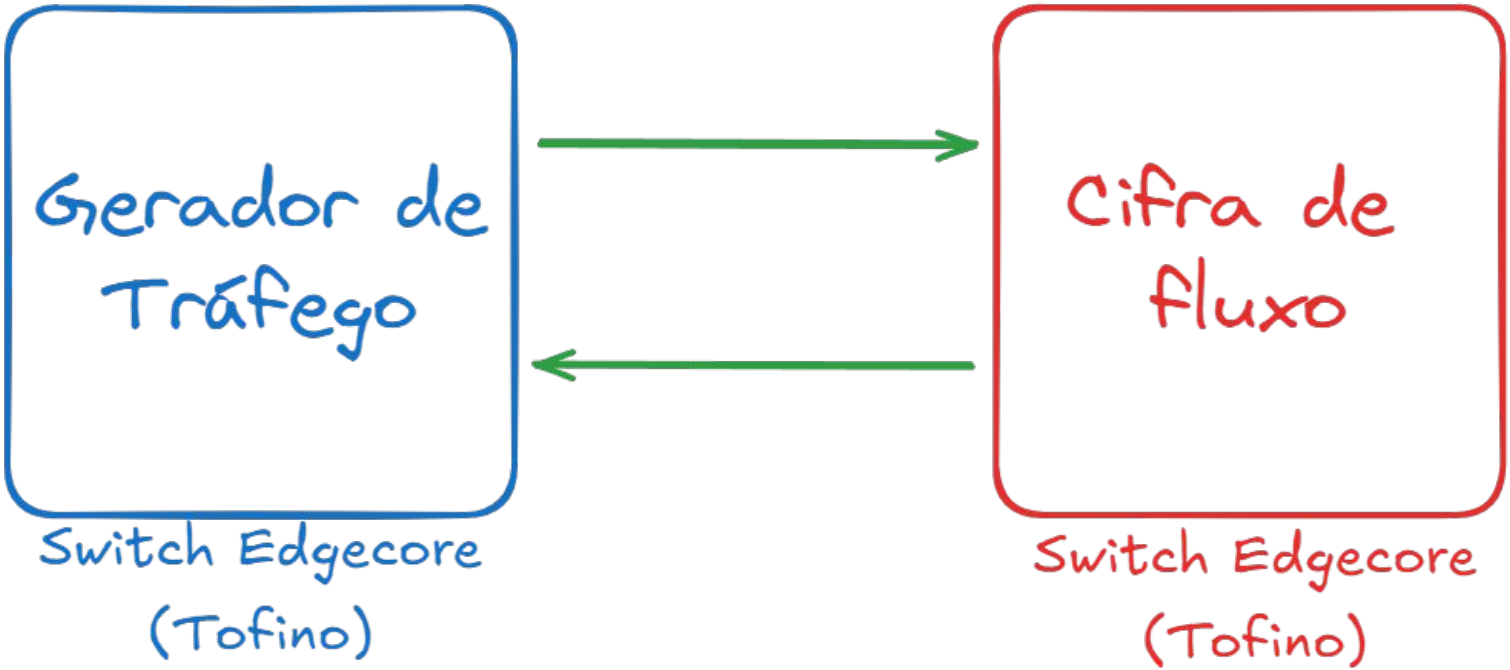
E quanto à vazão alcançada?



Uso dos recursos do Switch

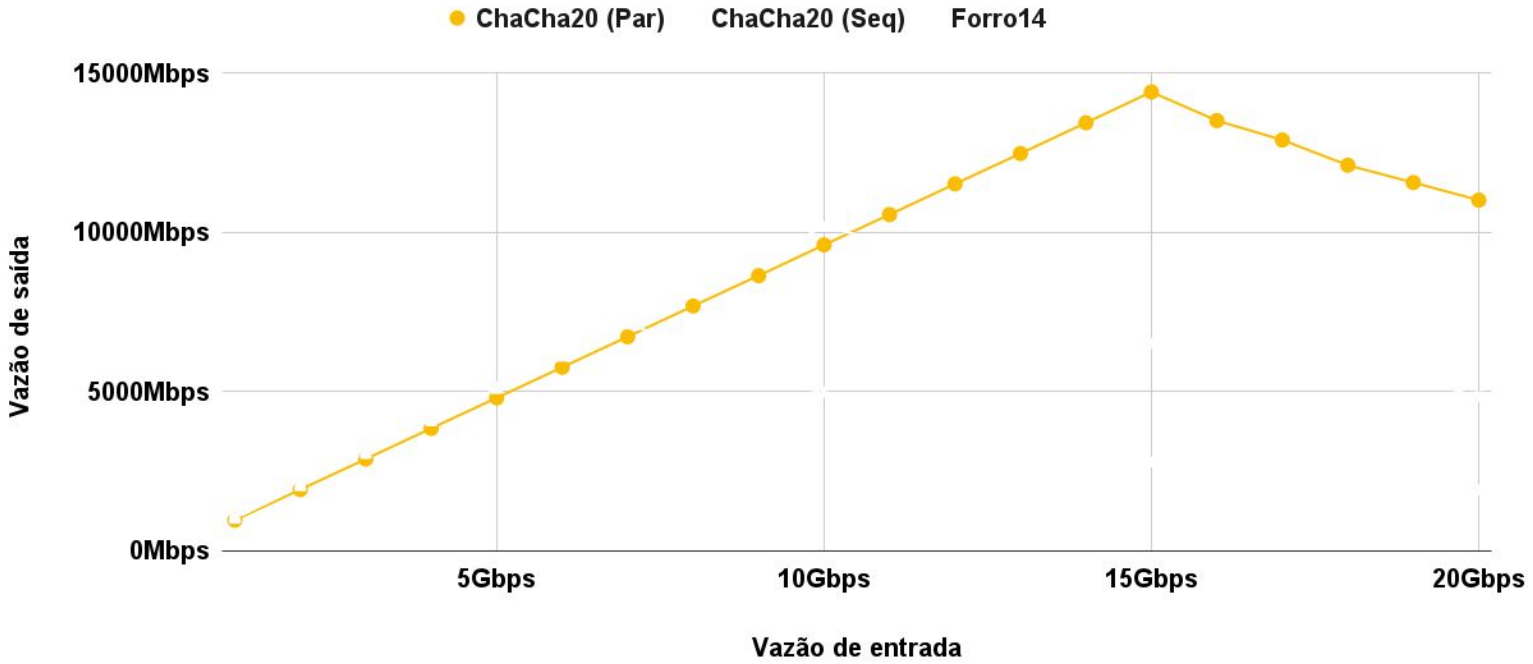
(quanto menor, melhor)

Avaliação: *Testbed*



Testbed utilizado

Avaliação: *Vazão*

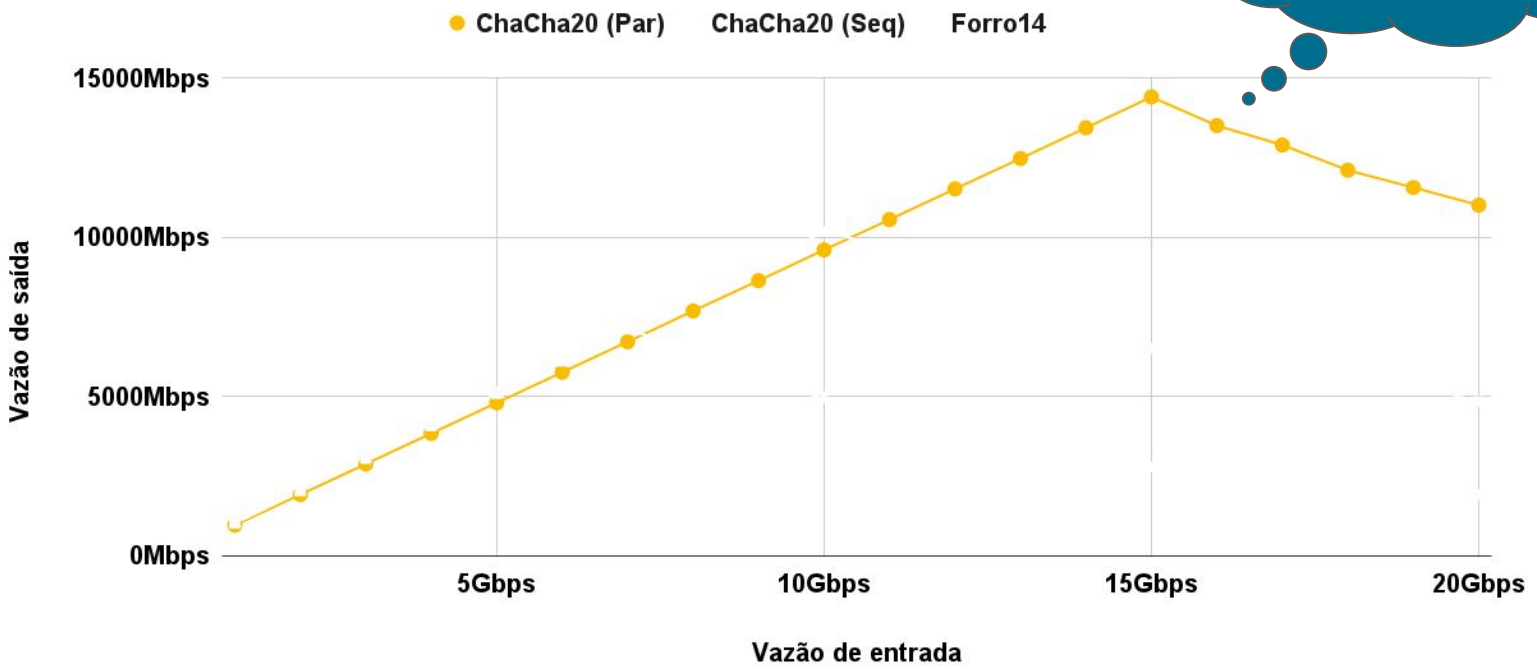


Vazão alcançada em Mbps

(Quanto mais, melhor)

Avaliação: *Vazão*

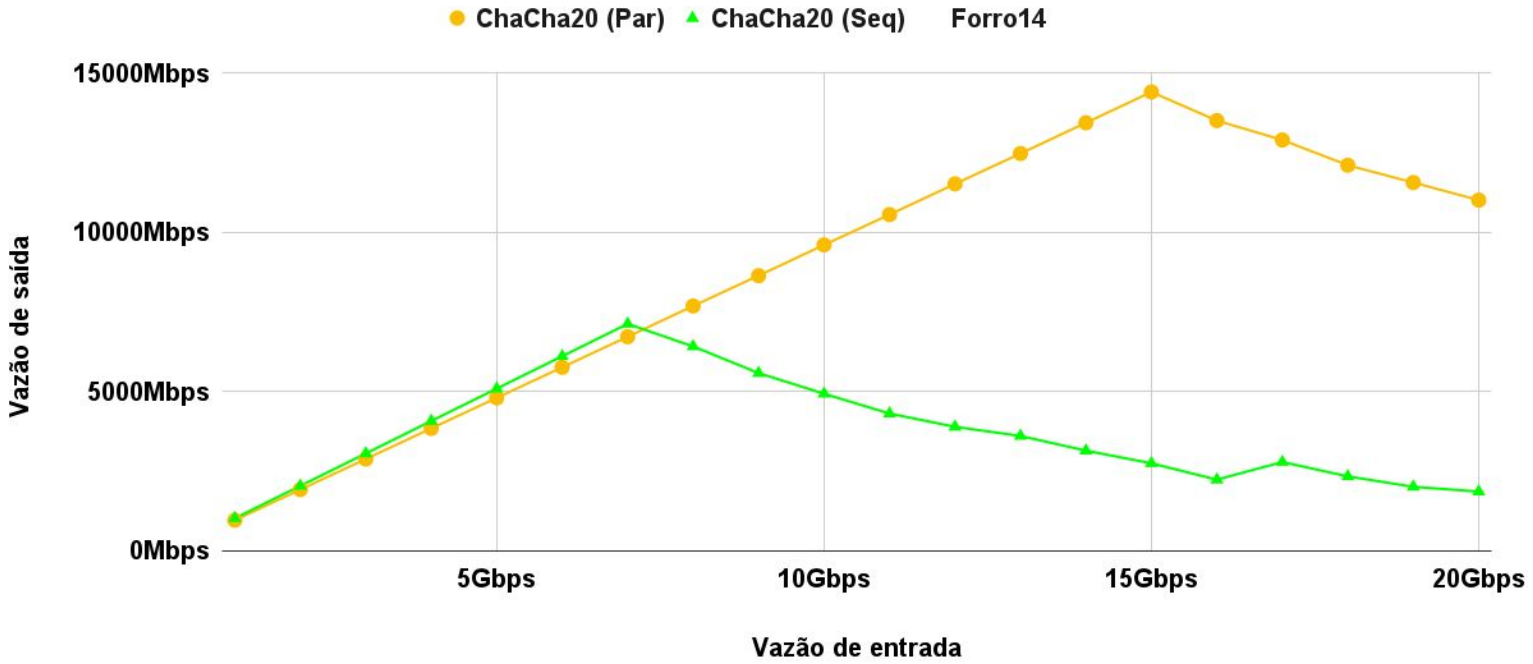
Mas e se eu tiver restrição de recursos?



Vazão alcançada em Mbps

(Quanto mais, melhor)

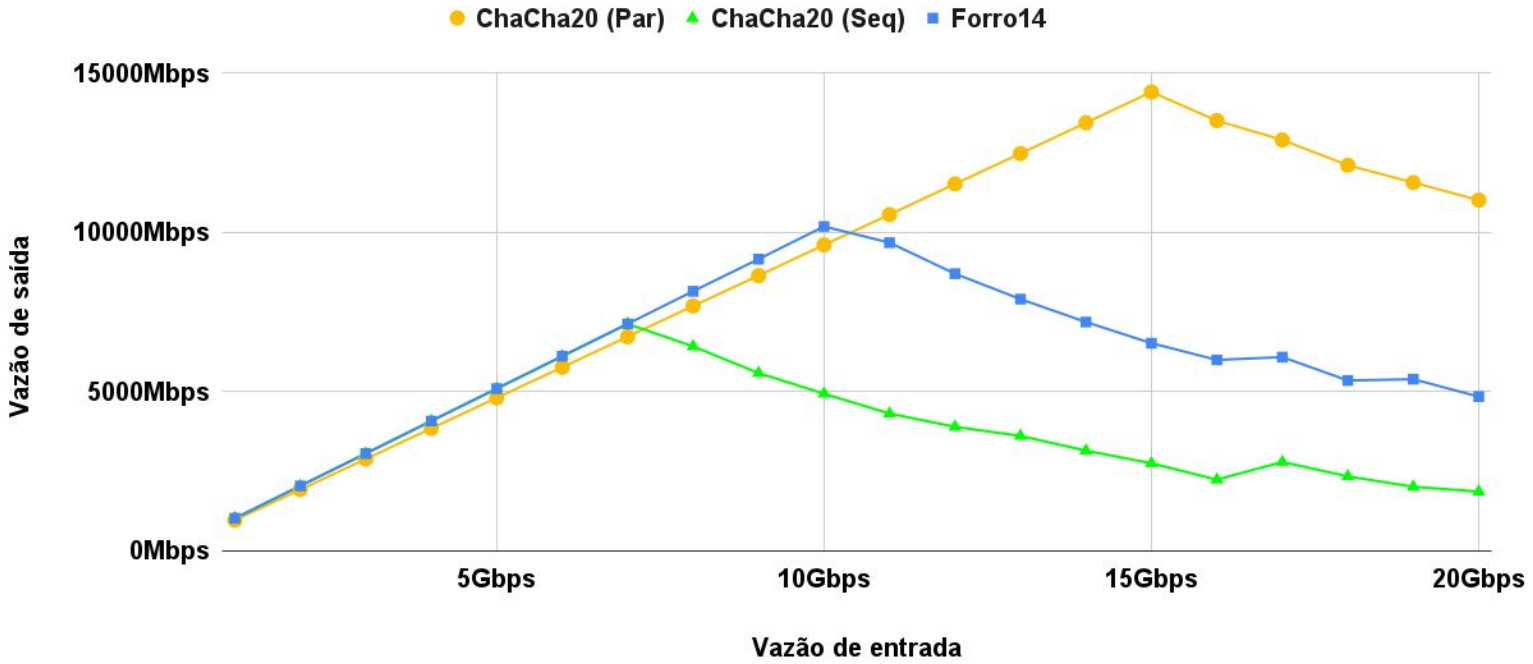
Avaliação: *Vazão*



Vazão alcançada em Mbps

(Quanto mais, melhor)

Avaliação: *Vazão*



Vazão alcançada em Mbps

(Quanto mais, melhor)

Avaliação: *Conclusões*

- Para cenários com restrição de recurso, o Forro14:
  - possui o mesmo uso de recursos do ChaCha20 sequencial com vazão máxima ~43% maior.
  - não utiliza a memória TCAM (escassa no *switch*)

Trabalhos futuros

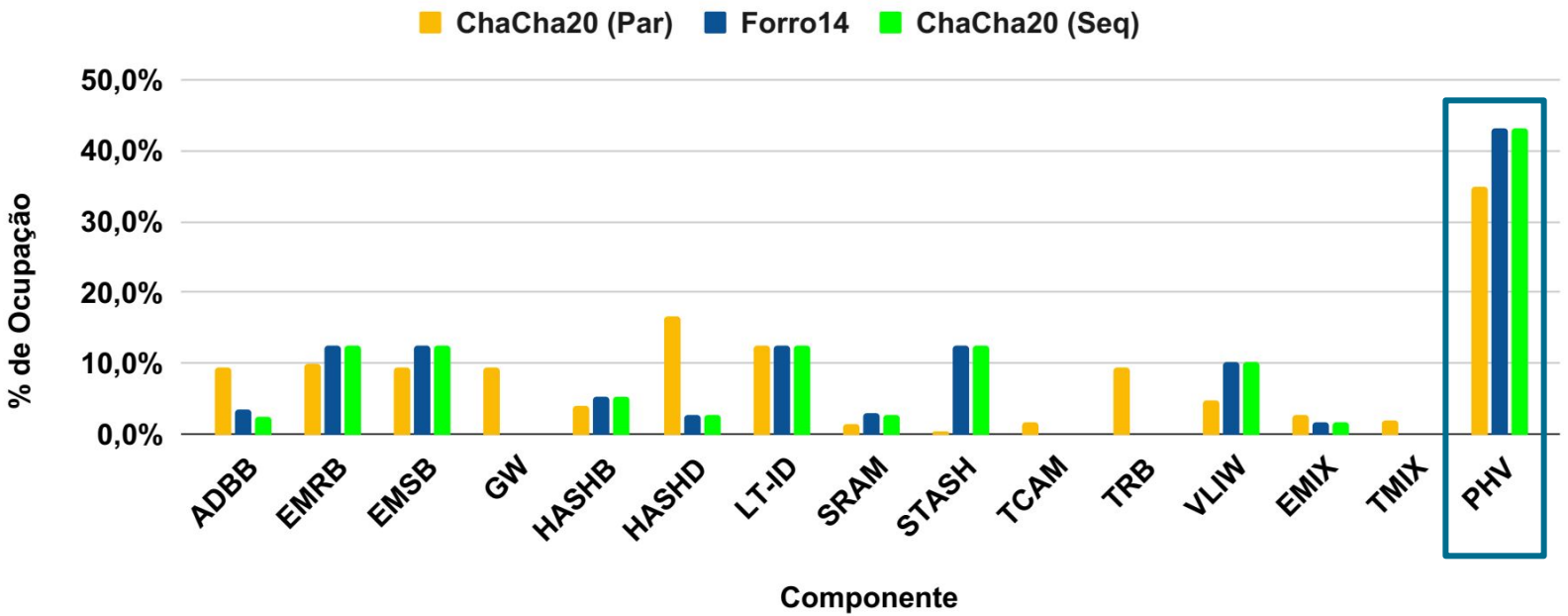
Trabalhos futuros

- Refatorar a implementação do Forro14 para reduzir o uso de *PHV*;

Trabalhos futuros

- Refatorar a implementação do Forro14 para reduzir o uso de *PHV*;
  - *PHV* (*Packet Header Vectors*): “alocação de memória” por estágio do pipeline.

Trabalhos futuros

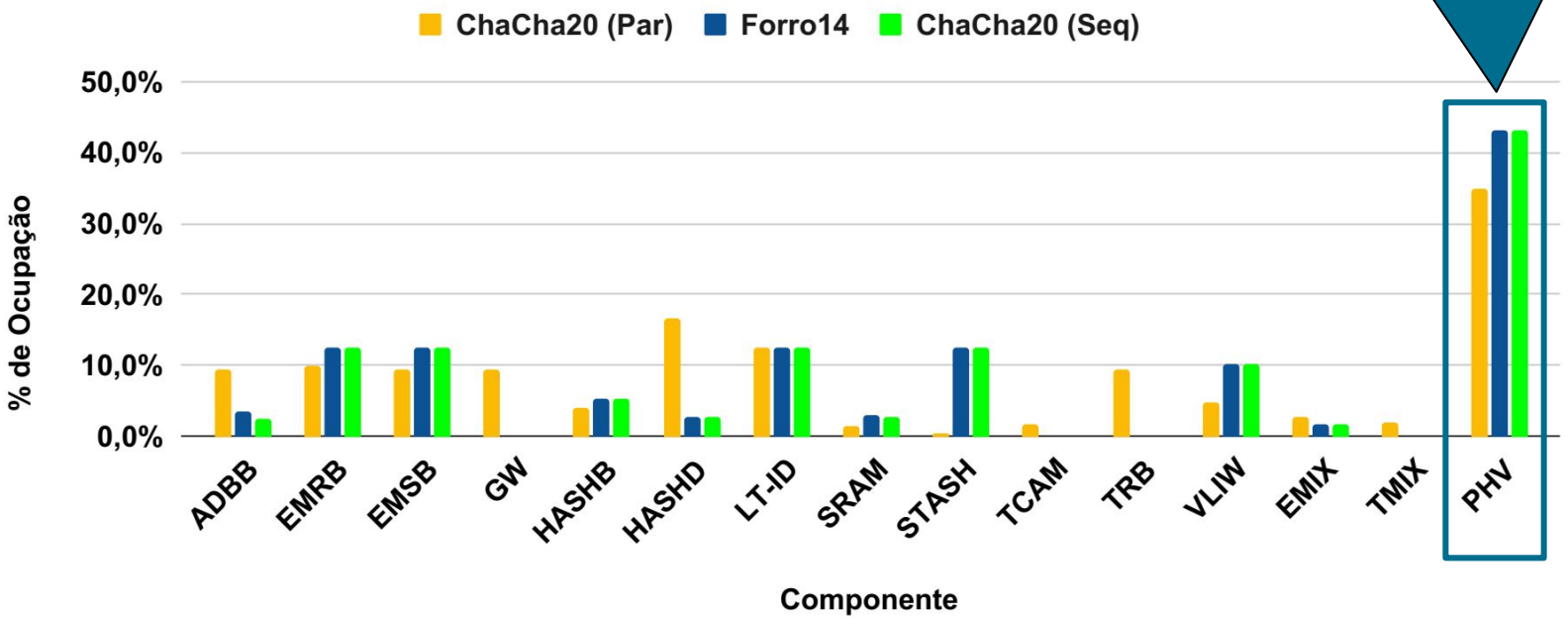


Uso dos recursos do Switch

(quanto menor, melhor)

Trabalhos futuros

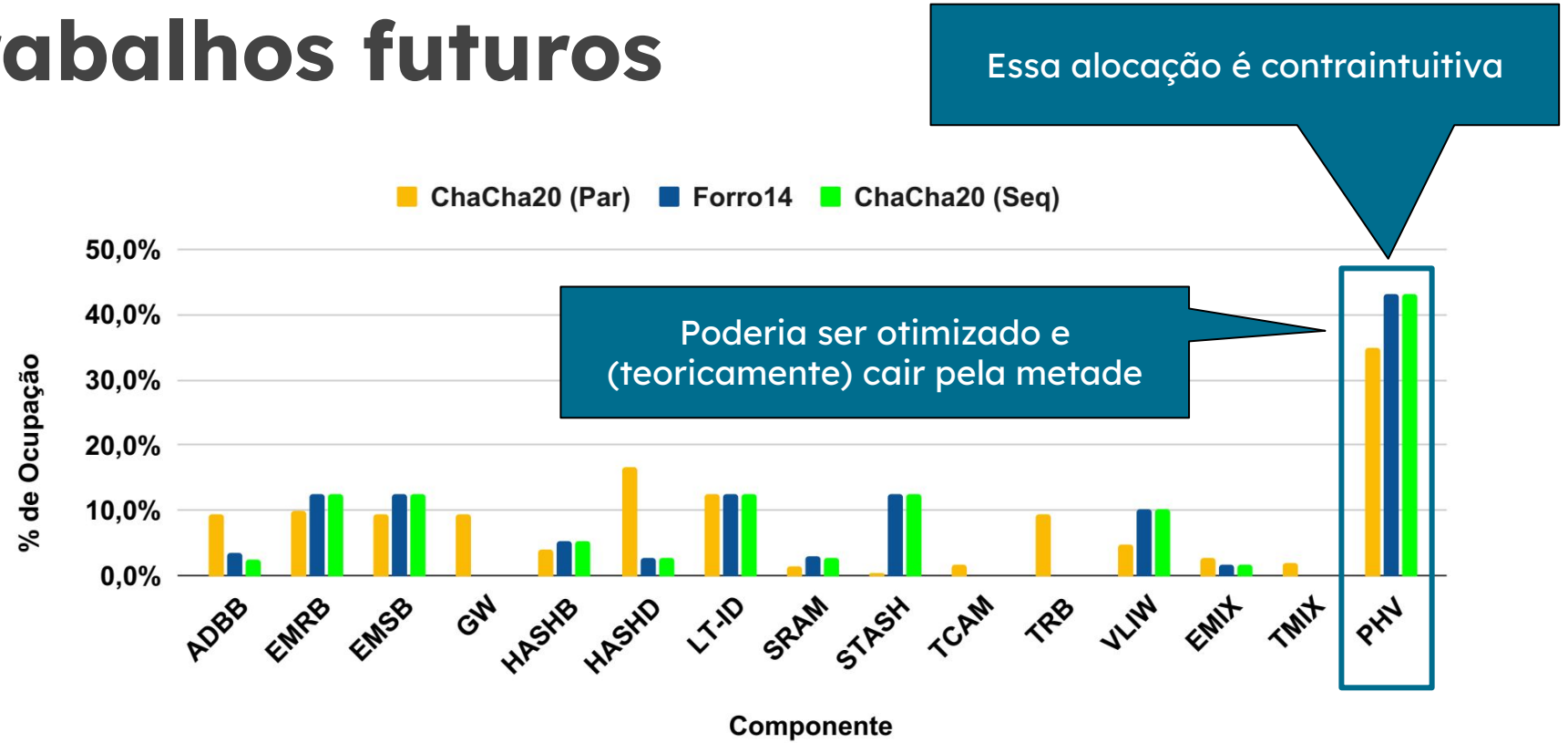
Essa alocação é contraintuitiva



Uso dos recursos do Switch

(quanto menor, melhor)

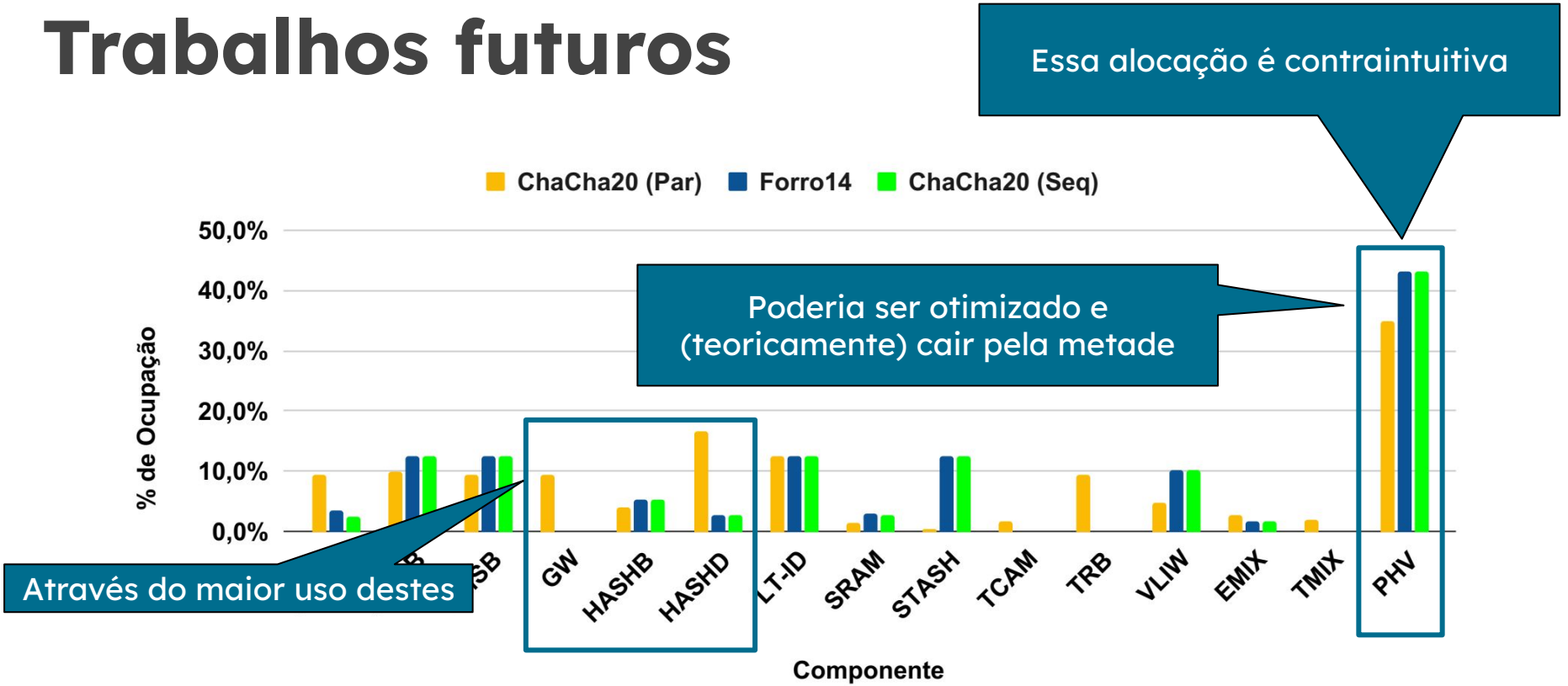
Trabalhos futuros



Uso dos recursos do Switch

(quanto menor, melhor)

Trabalhos futuros



Uso dos recursos do Switch

(quanto menor, melhor)

Trabalhos futuros

- Implementação e avaliação de uma versão paralelizada do Forro14

Trabalhos futuros

- Implementação e avaliação de uma versão paralelizada do Forro14
  - calcular mais de um estado por rodada;

Trabalhos futuros

- Implementar e avaliar as cifras com *AEAD* (Cifração Autenticada com Dados Associados)

Trabalhos futuros

- Implementar e avaliar as cifras com *AEAD* (Cifração Autenticada com Dados Associados)
  - XChaCha20-Poly1305 e XForro14-Poly1305.

ACKNOWLEDGMENTS & DISCLAIMER

This work has been performed within the framework of the FAPESP Engineering Research Center (ERC) Program under FAPESP grant agreement #2021/00199-8 (SMARTNESS).

Referências

- Coutinho, M., Passos, I., Vásquez, J. C. G., Sarkar, S., de Mendonça, F. L., de Sousa Jr, R. T., and Borges, F. (2023). **Latin dances reloaded: Improved cryptanalysis against salsa and chacha, and the proposal of forró**. *Journal of Cryptology*, 36(3):18.
- Peterson, L., Cascone, C., and Davie, B. (2021). **Software-Defined Networks: A Systems Approach**. *Systems Approach LLC*.
- Yoshinaka, Y., Takemasa, J., Koizumi, Y., and Hasegawa, T. (2022). **On implementing chacha on a programmable switch**. In *Proceedings of the 5th International Workshop on P4 in Europe*, pages 15–18.

Referências

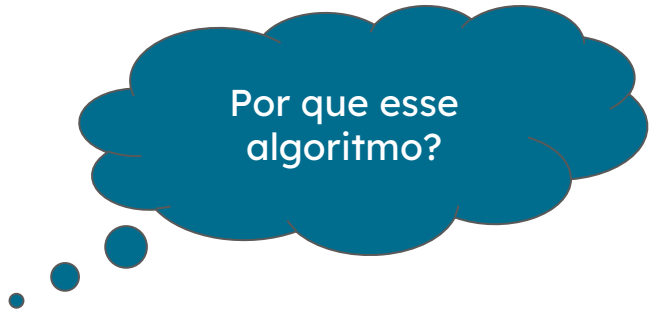
- Coutinho, M., Passos, I., and Borges, F. (2023a). **The design and implementation of xforró14-poly1305: a new authenticated encryption scheme**. In *Anais do XXIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 456–469, Porto Alegre, RS, Brasil. SBC.
- Chen, X. (2020). **Implementing aes encryption on programmable switches via scrambled lookup tables**. In *Proceedings of the Workshop on Secure Programmable Network Infrastructure*, pages 8–14.

Obrigado!

- Rodrigo Alexander de Andrade Pierini
- rpierini@unicamp.br

Backup

- Cifração em plano de dados programável tem alto impacto de desempenho e recursos
  - AES [Chen, 2020];
  - ChaCha20 [Yoshinaka *et al*, 2022]

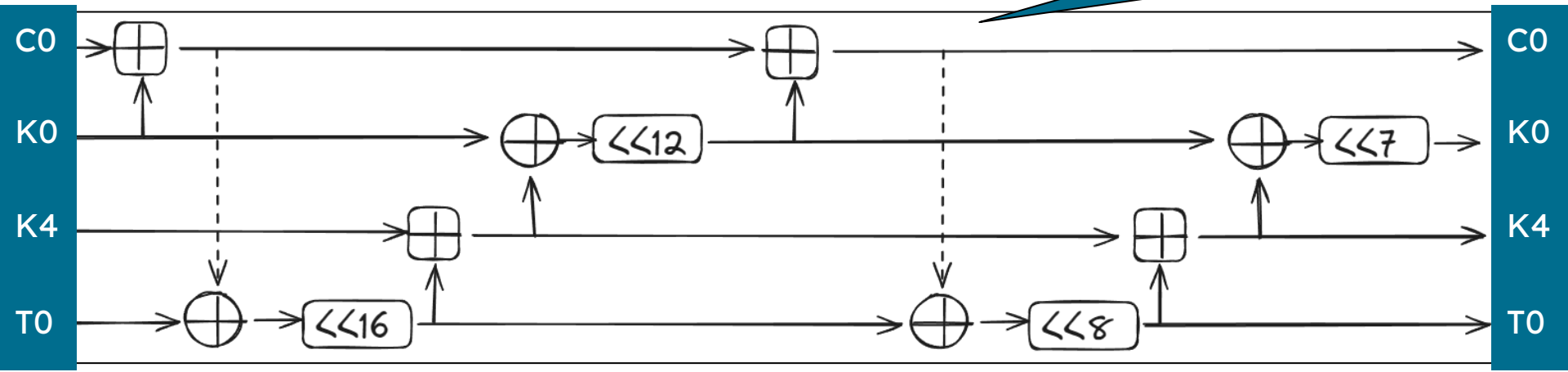


Backup

- Cifração em plano de dados programável tem alto impacto de desempenho e recursos
  - AES [Chen, 2020];
  - ChaCha20 [Yoshinaka *et al*, 2022]
    - Usado no TLS1.3
    - Usado como DRNG no Kernel Linux

Backup

1° Quarter-Round (QR0)



⊕ adição mod 2^32

⊕ bitwise XOR

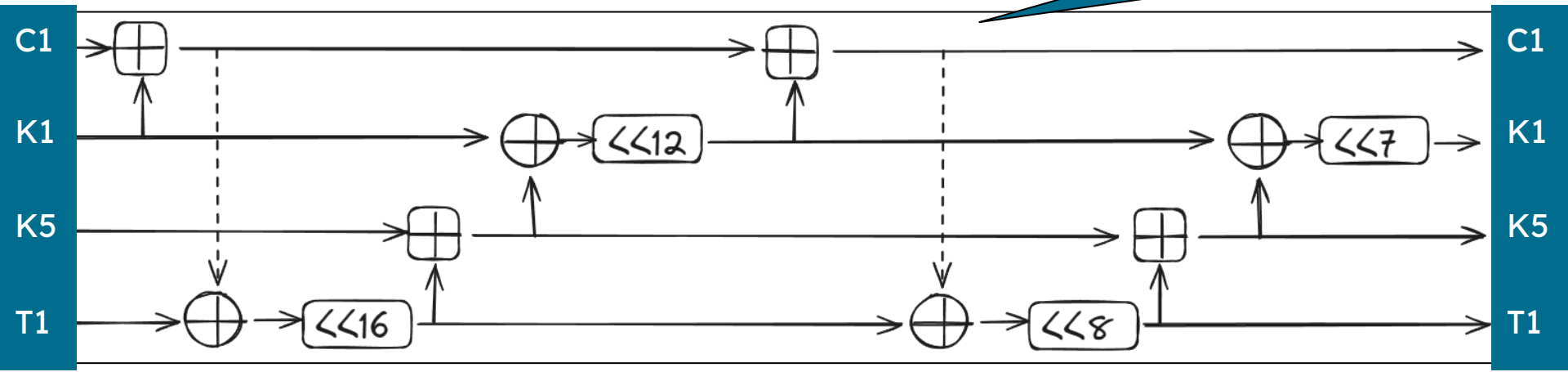
<<<N Rotação circular à esquerda

	0	1	2	3
A	c0	c1	c2	c3
B	k0	k1	k2	k3
C	k4	k5	k6	k7
D	t0	t1	n0	n1

c = constante
k = chave
t = contador
n = nonce

Backup

2º Quarter-Round (QR1)



⊕ adição mod 2^32

⊕ bitwise XOR

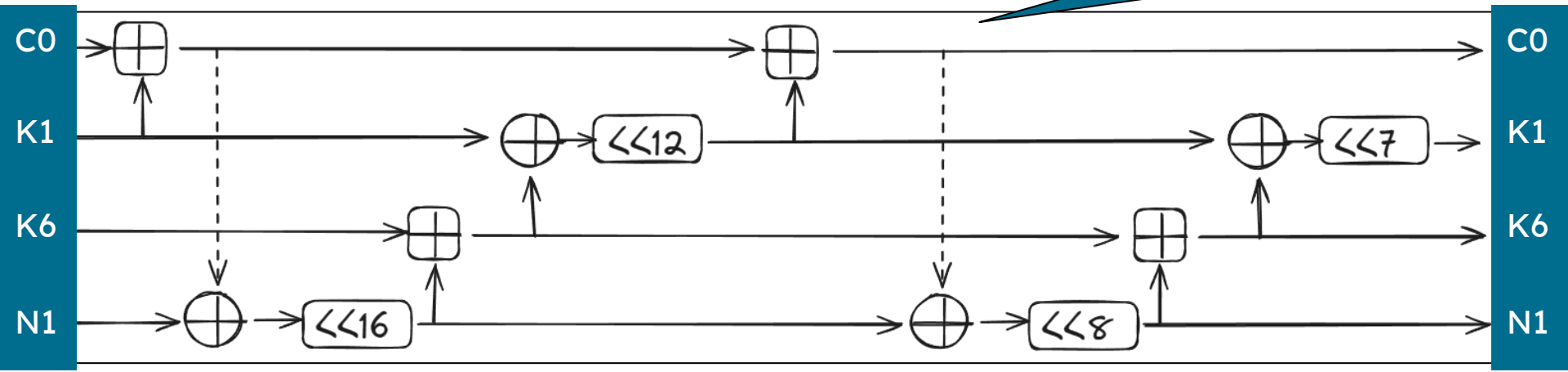
<<<N Rotação circular à esquerda

	0	1	2	3
A	c0	c1	c2	c3
B	k0	k1	k2	k3
C	k4	k5	k6	k7
D	t0	t1	n0	n1

c = constante
k = chave
t = contador
n = nonce

Backup

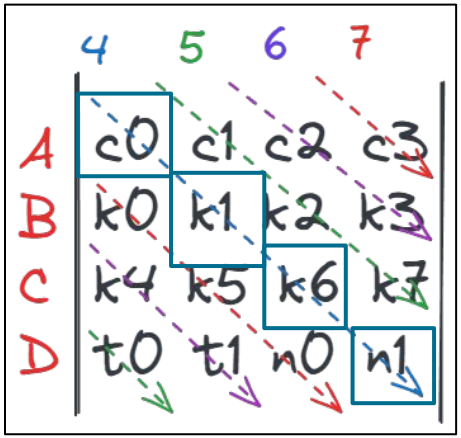
5º Quarter-Round (QR4)



⊞ adição mod 2^32

⊕ bitwise XOR

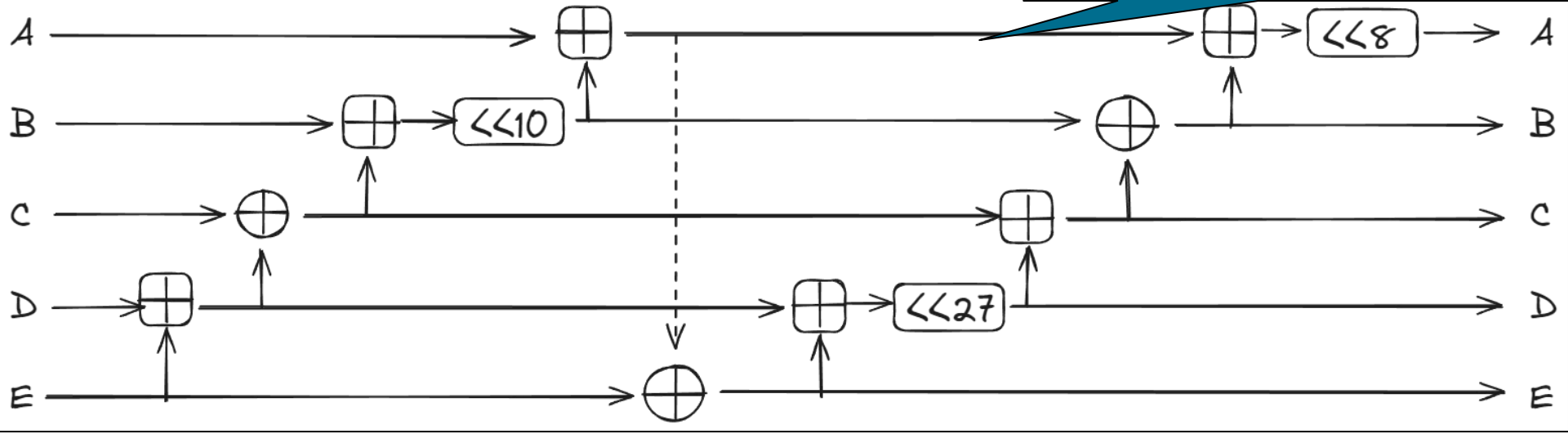
<<<N Rotação circular à esquerda



c = constante
k = chave
t = contador
n = nonce

Backup

Quarter-Round Function (QR)



⊞ adição mod 2^32

⊕ bitwise XOR

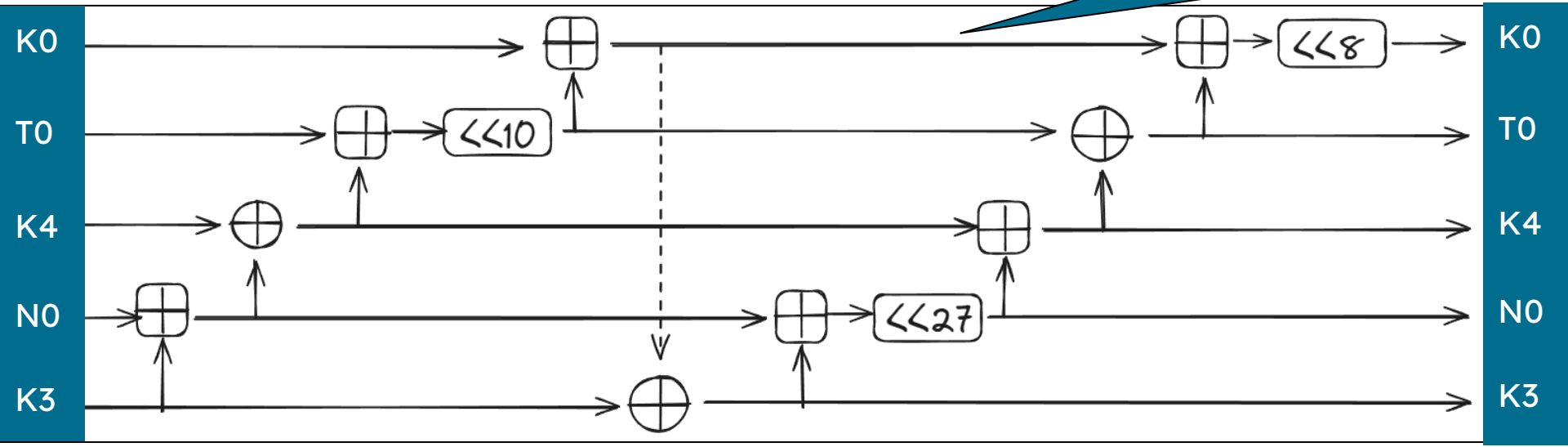
<<<N Rotação circular à esquerda

	0	1	2	3
A	k0	k1	k2	k3
B	t0	t1	c0	c1
C	k4	k5	k6	k7
D	n0	n1	c2	c3

c = constante
k = chave
t = contador
n = nonce

Backup

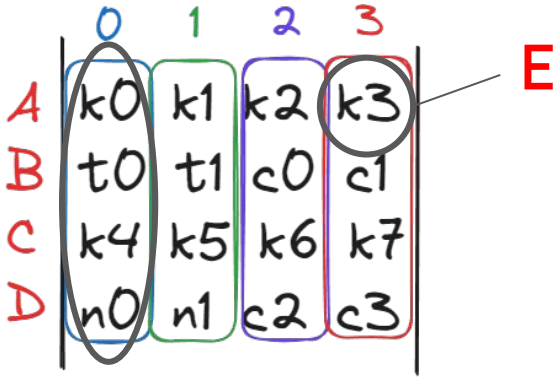
1º Quarter-Round (QR0)



⊞ adição mod 2^32

⊕ bitwise XOR

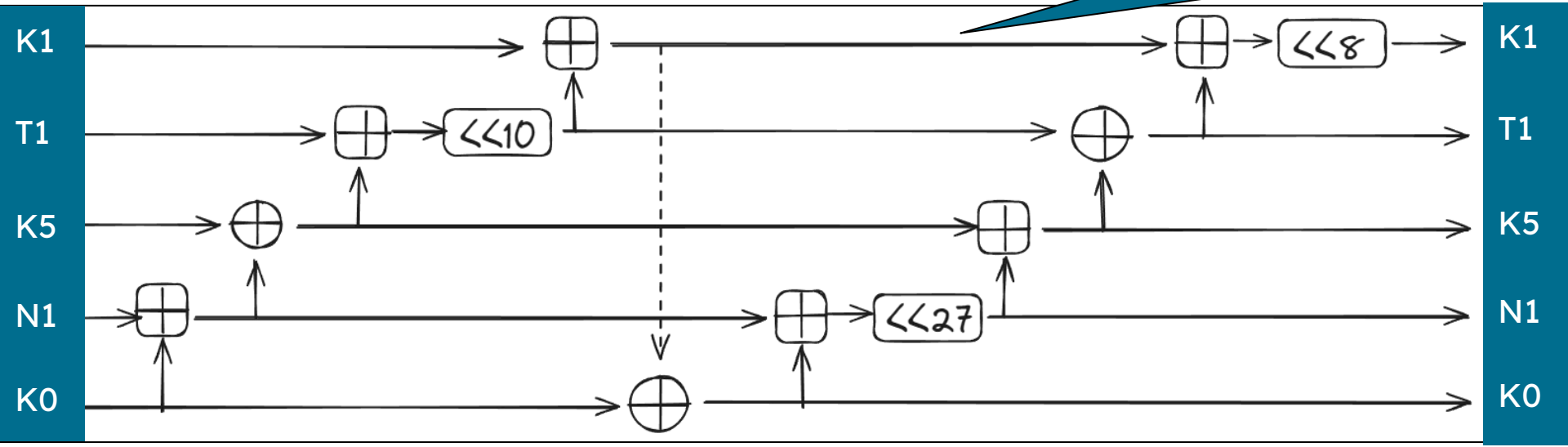
<<<N Rotação circular à esquerda



c = constante
k = chave
t = contador
n = nonce

Backup

2º Quarter-Round (QR1)



⊕ adição mod 2^32

⊕ bitwise XOR

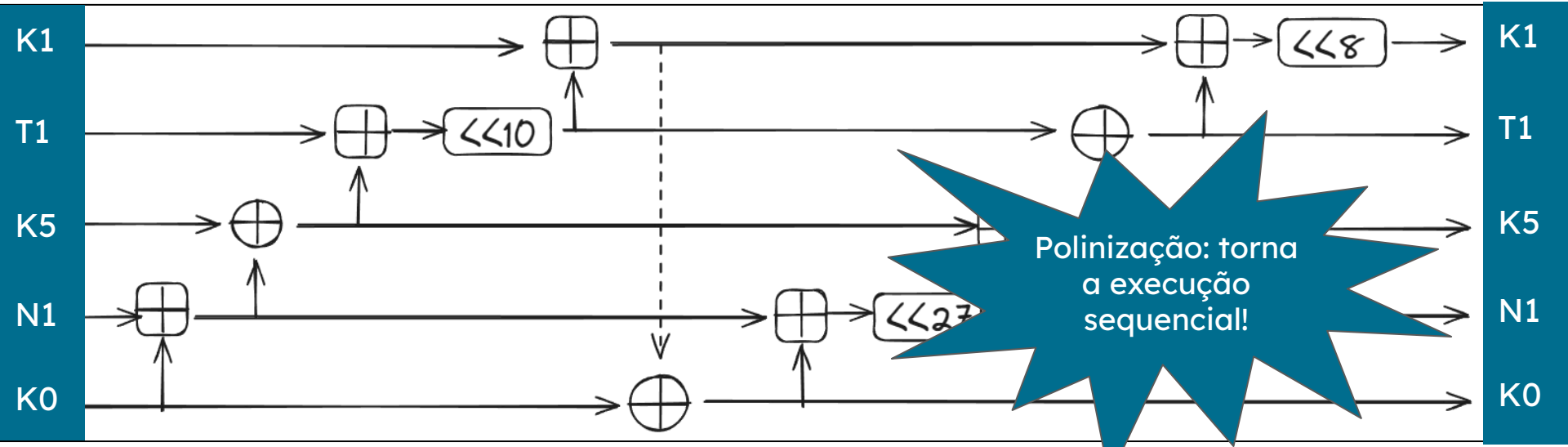
<<<N Rotação circular à esquerda

E

	0	1	2	3
A	k0	k1	k2	k3
B	t0	t1	c0	c1
C	k4	k5	k6	k7
D	n0	n1	c2	c3

c = constante
k = chave
t = contador
n = nonce

Backup



adição mod 2^32

bitwise XOR

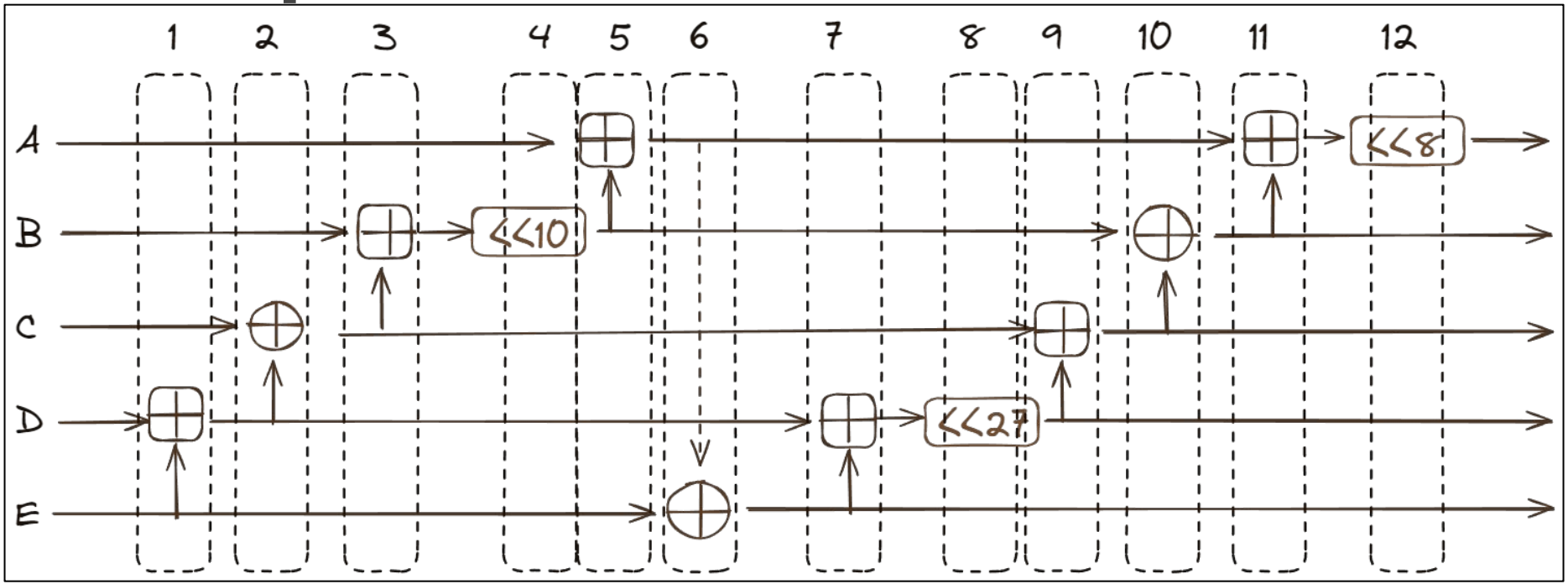
<<<N Rotação circular à esquerda

E

	0	1	2	3
A	k0	k1	k2	k3
B	t0	t1	c0	c1
C	k4	k5	k6	k7
D	n0	n1	c2	c3

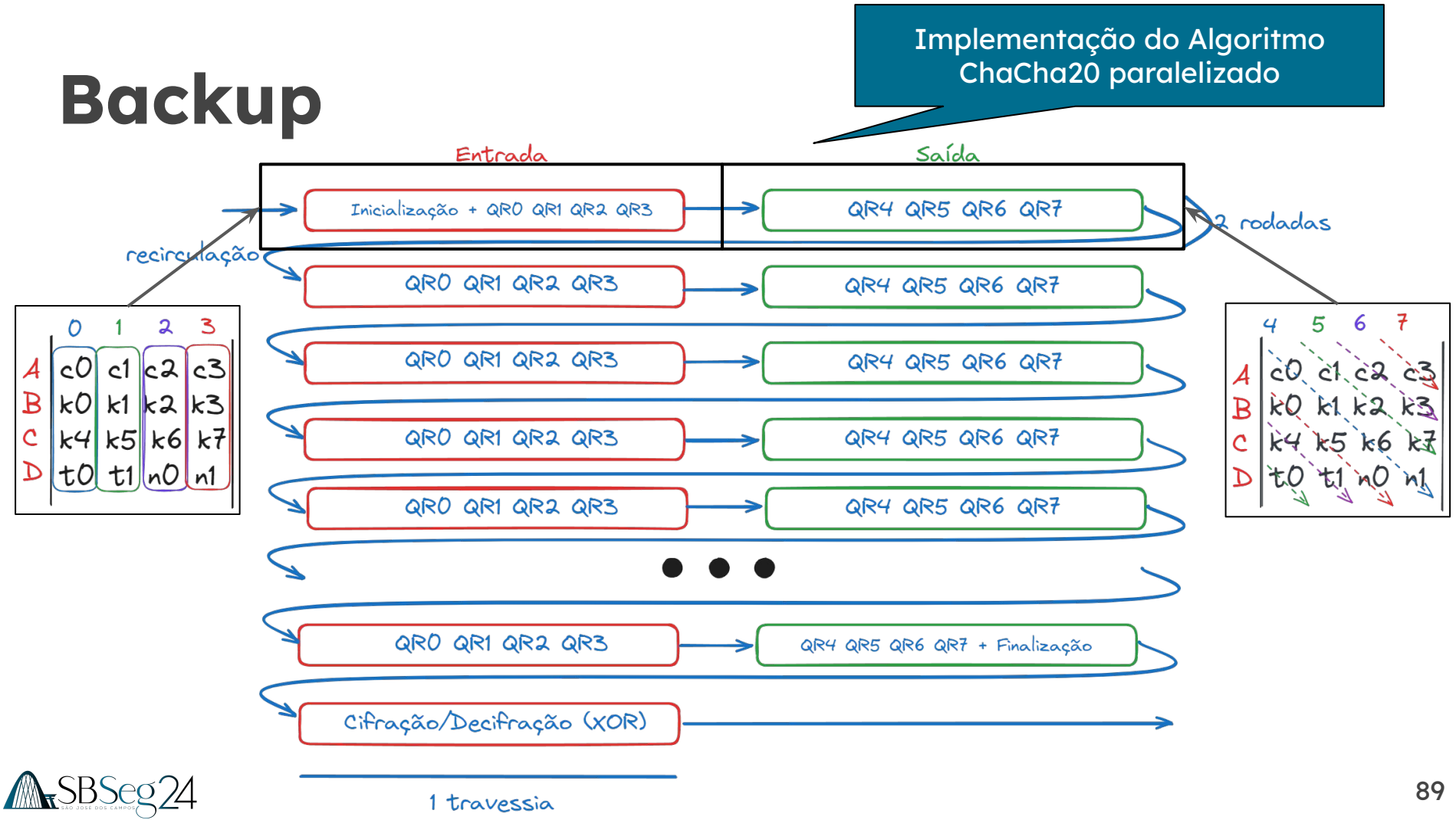
c = constante
k = chave
t = contador
n = nonce

Backup

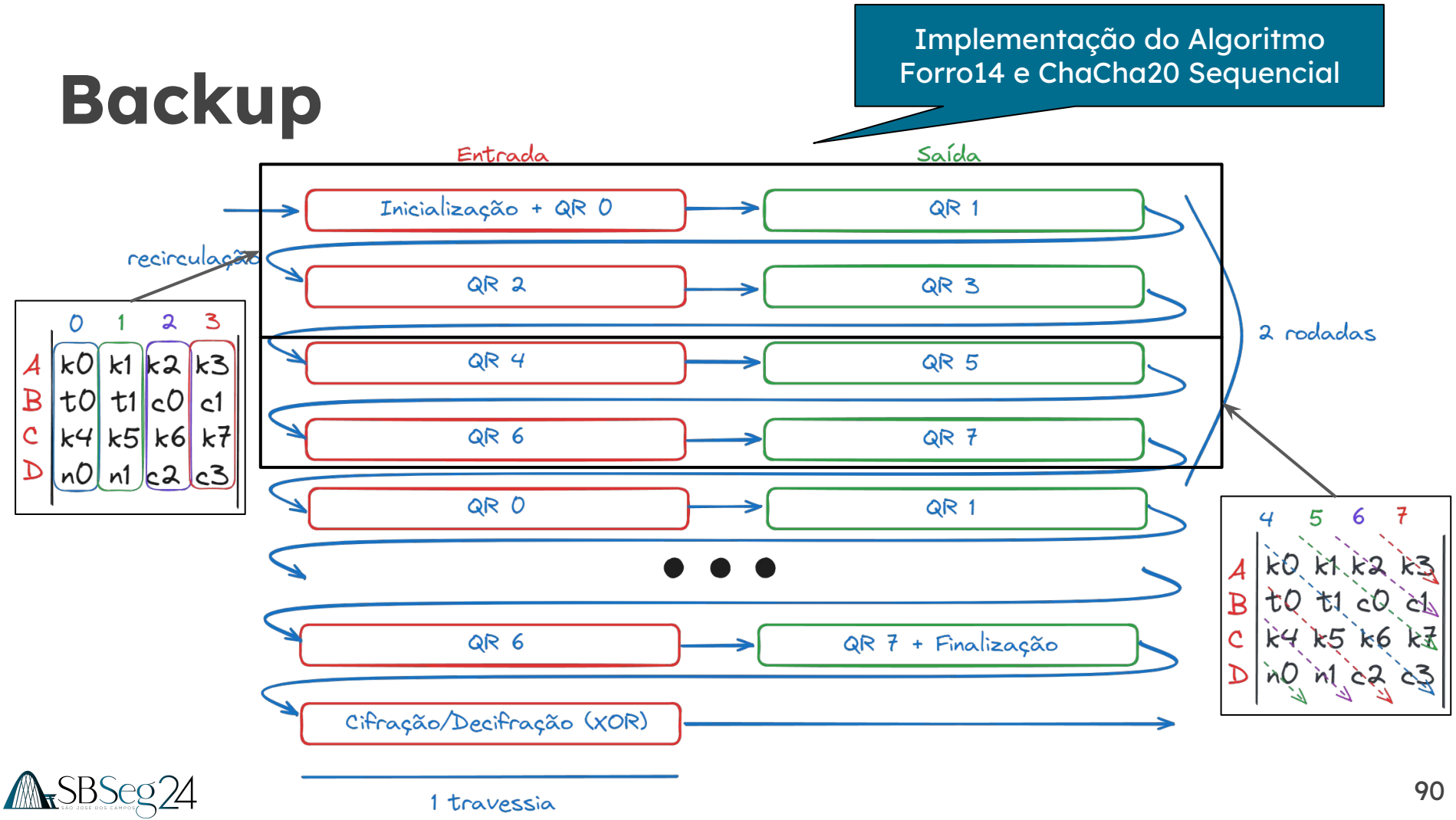


Divisão dos QRs em 12 estágios

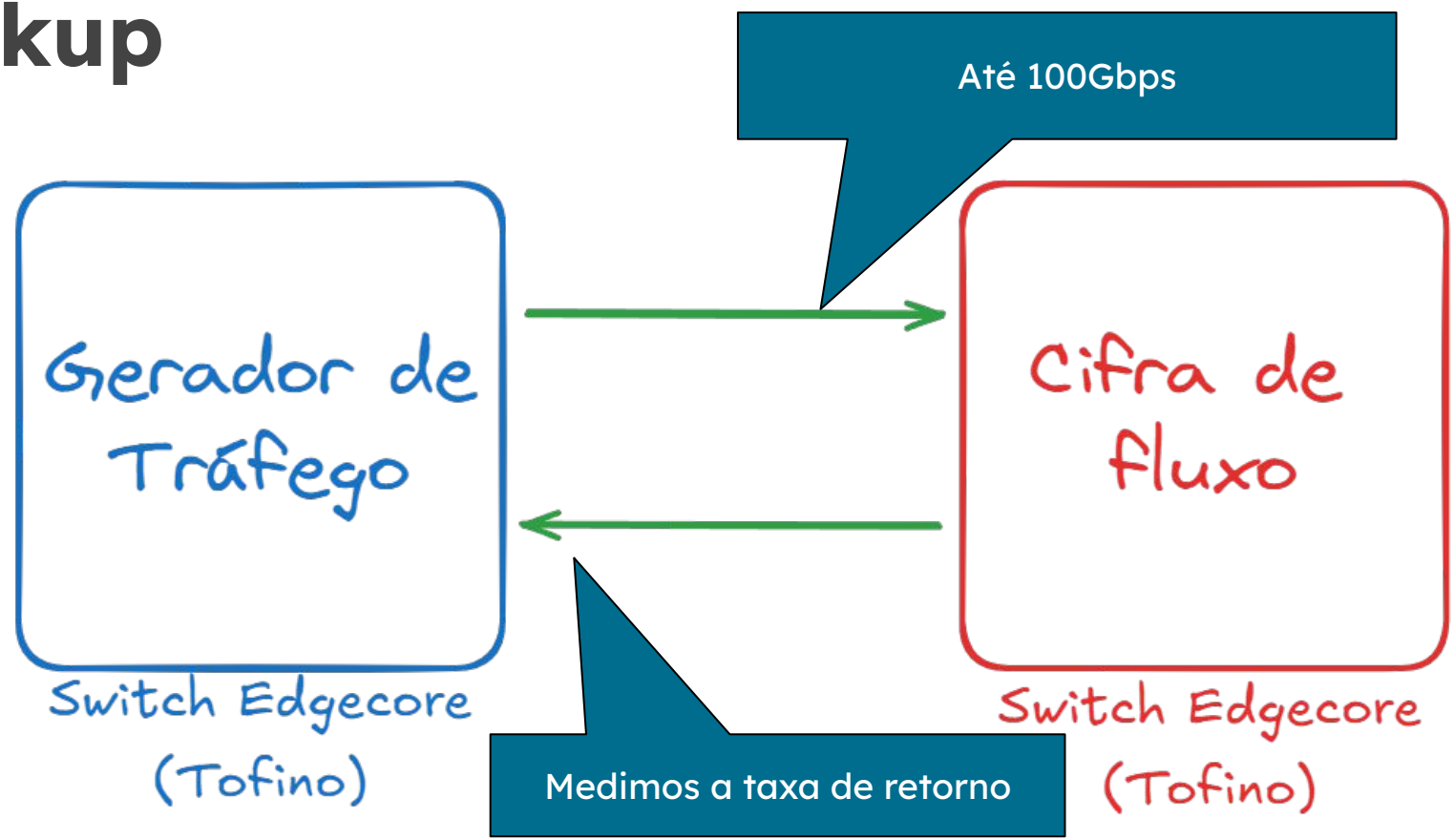
Backup


Implementação do Algoritmo ChaCha20 paralelizado
Entrada
Saída
Inicialização + QR0 QR1 QR2 QR3
QR4 QR5 QR6 QR7
2 rodadas
recirculação
QR0 QR1 QR2 QR3
QR4 QR5 QR6 QR7
QR0 QR1 QR2 QR3
QR4 QR5 QR6 QR7
QR0 QR1 QR2 QR3
QR4 QR5 QR6 QR7
QR0 QR1 QR2 QR3
QR4 QR5 QR6 QR7
QR0 QR1 QR2 QR3
QR4 QR5 QR6 QR7 + Finalização
Cifração/Decifração (XOR)
1 travessia
0 1 2 3
A c0 c1 c2 c3
B k0 k1 k2 k3
C k4 k5 k6 k7
D t0 t1 n0 n1
4 5 6 7
A c0 c1 c2 c3
B k0 k1 k2 k3
C k4 k5 k6 k7
D t0 t1 n0 n1

Backup

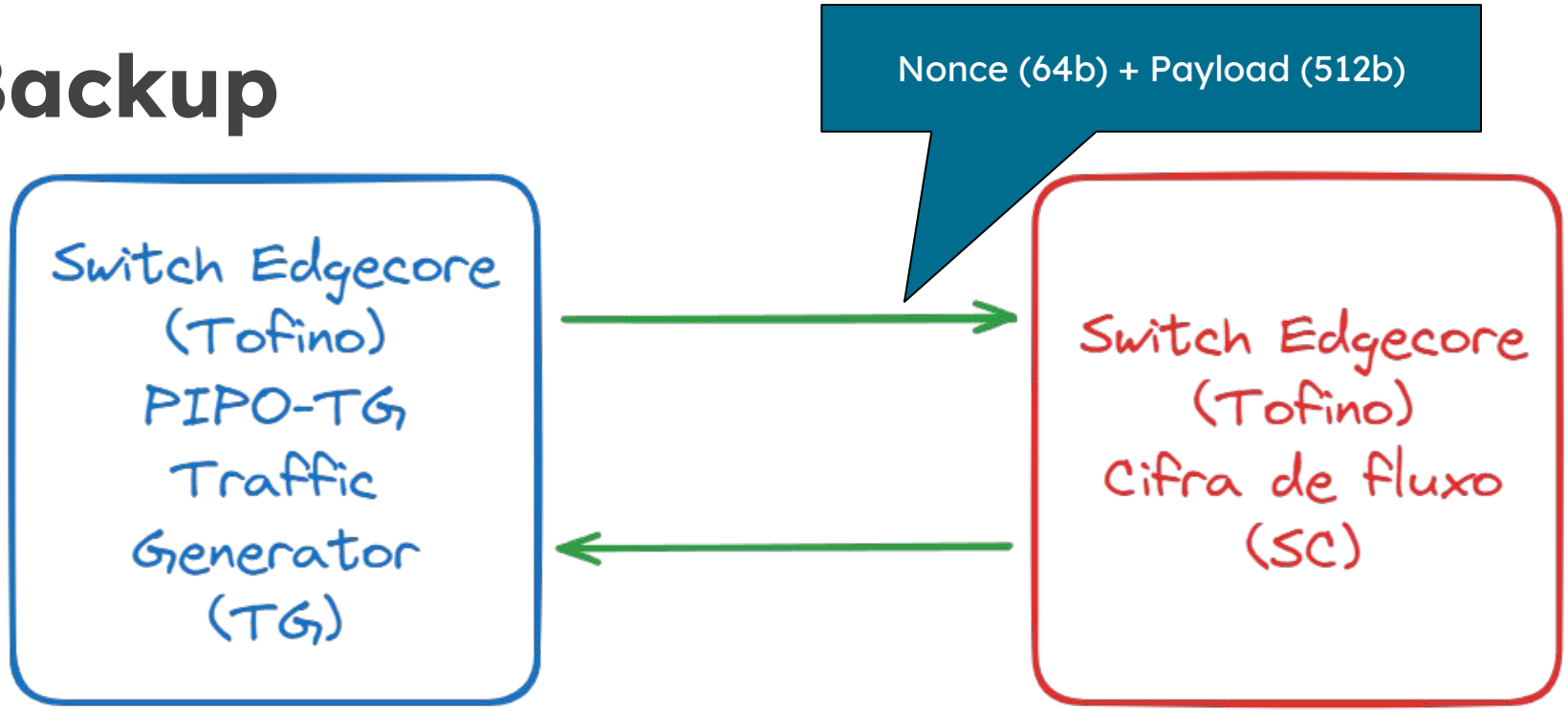

Implementação do Algoritmo
Forro14 e ChaCha20 Sequencial
Entrada
Saída
Inicialização + QR 0
QR 1
recirculação
QR 2
QR 3
QR 4
QR 5
QR 6
QR 7
2 rodadas
QR 0
QR 1
QR 6
QR 7 + Finalização
Cifração/Decifração (XOR)
1 travessia
0 1 2 3
A k0 k1 k2 k3
B t0 t1 c0 c1
C k4 k5 k6 k7
D n0 n1 c2 c3
4 5 6 7
A k0 k1 k2 k3
B t0 t1 c0 c1
C k4 k5 k6 k7
D n0 n1 c2 c3

Backup



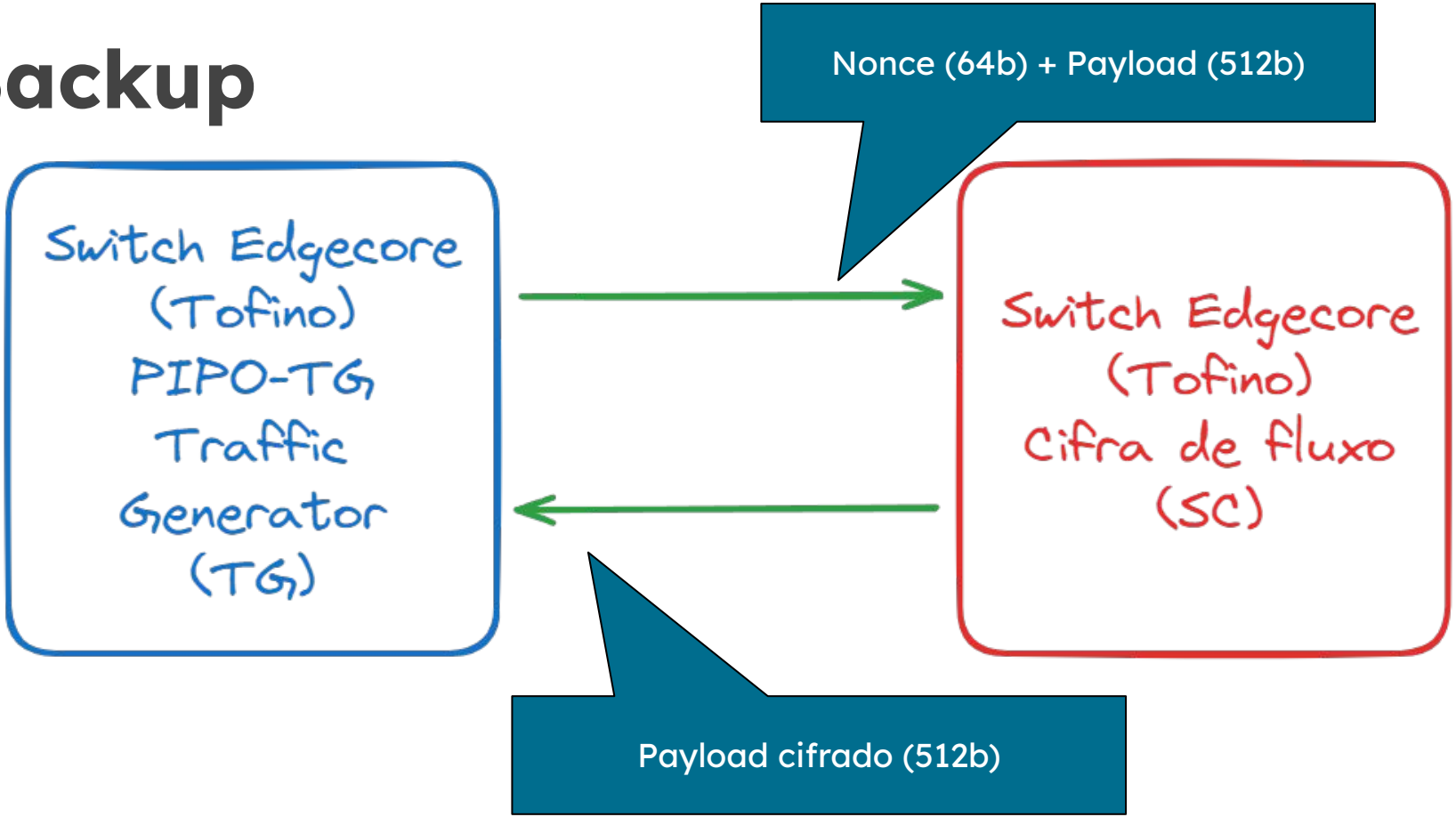
Testbed utilizado

Backup



Testbed utilizado

Backup



Testbed utilizado

Backup

	Taxa de entrada											
Algoritmo	**1Gbps**	**2Gbps**	**3Gbps**	**4Gbps**	**5Gbps**	**6Gbps**	**7Gbps**	**8Gbps**	**9Gbps**	**10Gbps**	**11Gbps**	**12Gbps**
Forro14	1019	2038	3055	4076	5093	6110	7127	8152	9161	10186	9678	8701
ChaCha20 (Seq)	1019	2038	3055	4076	5093	6110	7127	6418	5581	4932	4310	3893
ChaCha20 (Par)	961	1922	2881	3844	4803	5763	6722	7689	8640	9607	10559	11523
	Taxa de entrada											
Algoritmo	**13Gbps**	**14Gbps**	**15Gbps**	**16Gbps**	**17Gbps**	**18Gbps**	**19Gbps**	**20Gbps**	**25Gbps**	**40Gbps**	**50Gbps**	**100Gbps**
Forro14	7904	7184	6520	5994	6083	5349	5388	4842	3594	1313	912	46
ChaCha20 (Seq)	3606	3145	2747	2233	2787	2336	2009	1860	1163	268	155	2,6
ChaCha20 (Par)	12477	13442	14410	13511	12901	12109	11565	11013	6971	2862	2789	476

Vazão alcançada em Mbps

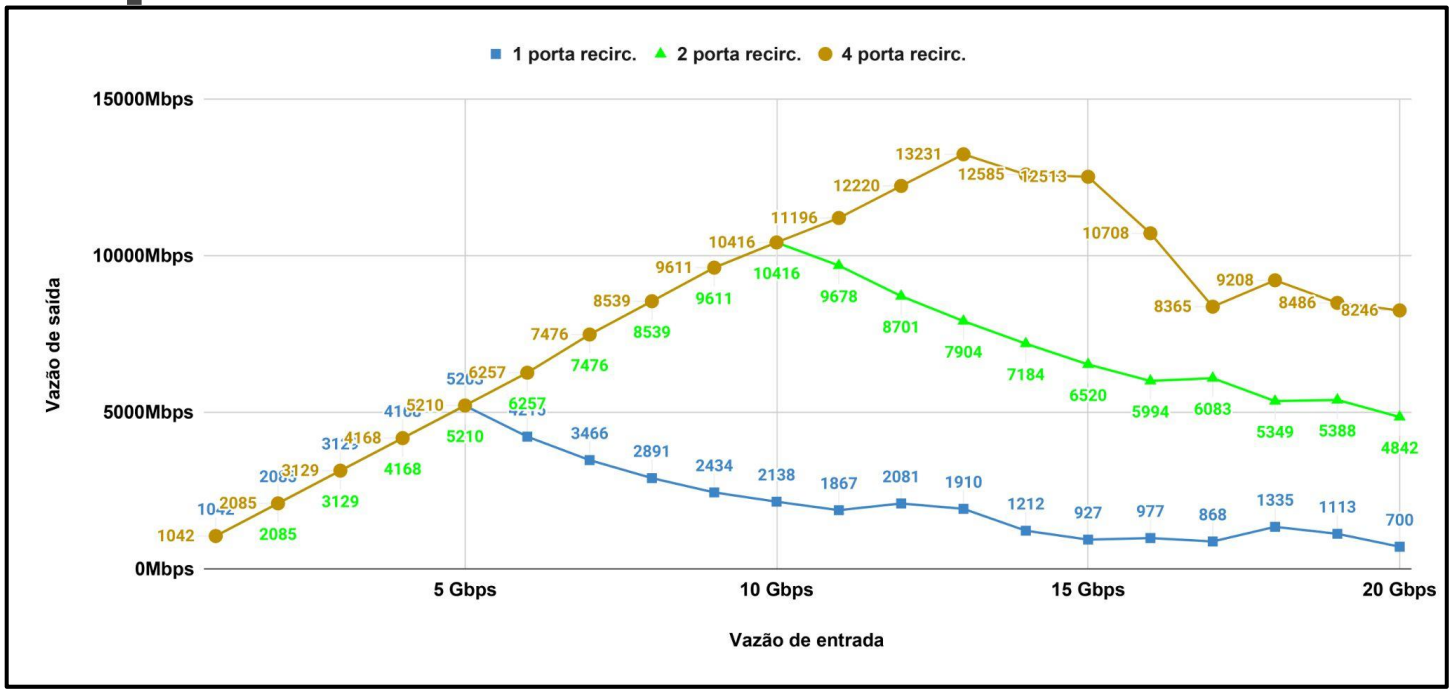
(Quanto mais, melhor)

Backup

Componente	ChaCha20 (Par)	Forro14	ChaCha20 (Seq)
ADBB	9,4%	3,6%	2,6%
EMRB	9,9%	12,5%	12,5%
EMSB	9,4%	12,5%	12,5%
GW	9,4%	0,0%	0,0%
HASHB	4,0%	5,4%	5,4%
HASHD	16,7%	2,8%	2,8%
LT-ID	12,5%	12,5%	12,5%
SRAM	1,4%	2,9%	2,7%
STASH	0,5%	12,5%	12,5%
TCAM	1,7%	0,0%	0,0%
TRB	9,4%	0,0%	0,0%
VLIW	4,7%	10,2%	10,2%
EMIX	2,8%	1,8%	1,8%
TMIX	2,1%	0,0%	0,0%
PHV	34,9%	43,1%	43,1%

Ocupação de recursos do *switch*

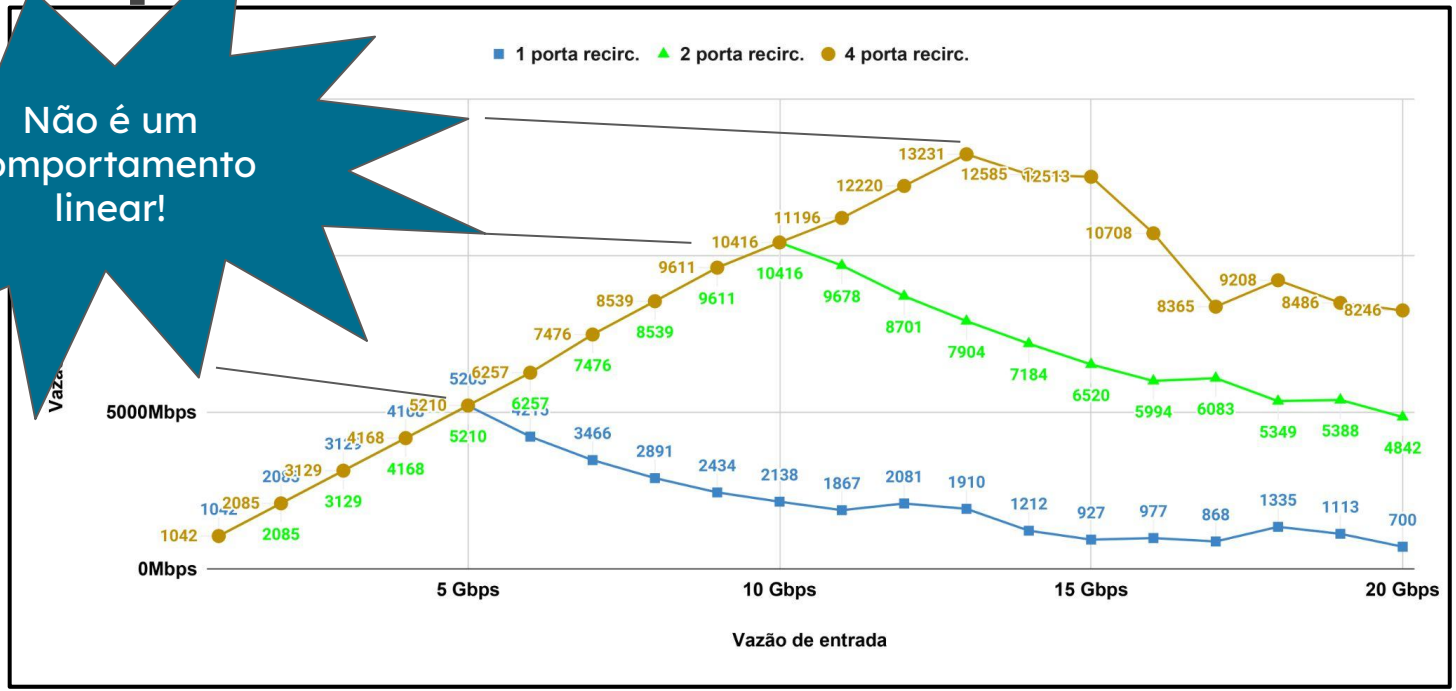
Backup



Vazão do Forro14 utilizando diferentes qtdes de portas para recirculação

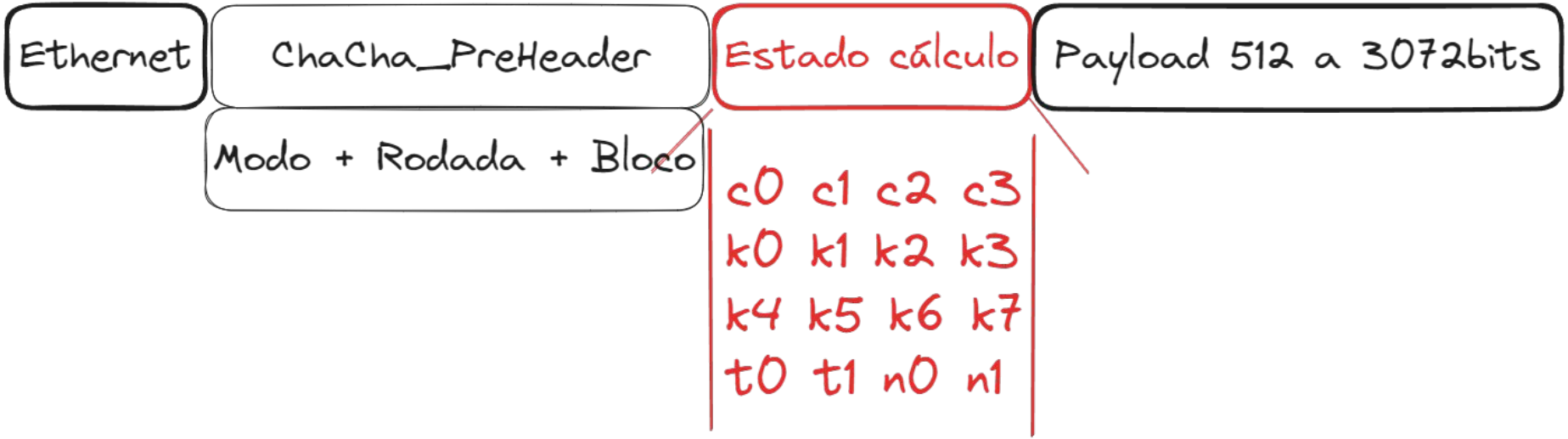
Backup

Não é um comportamento linear!

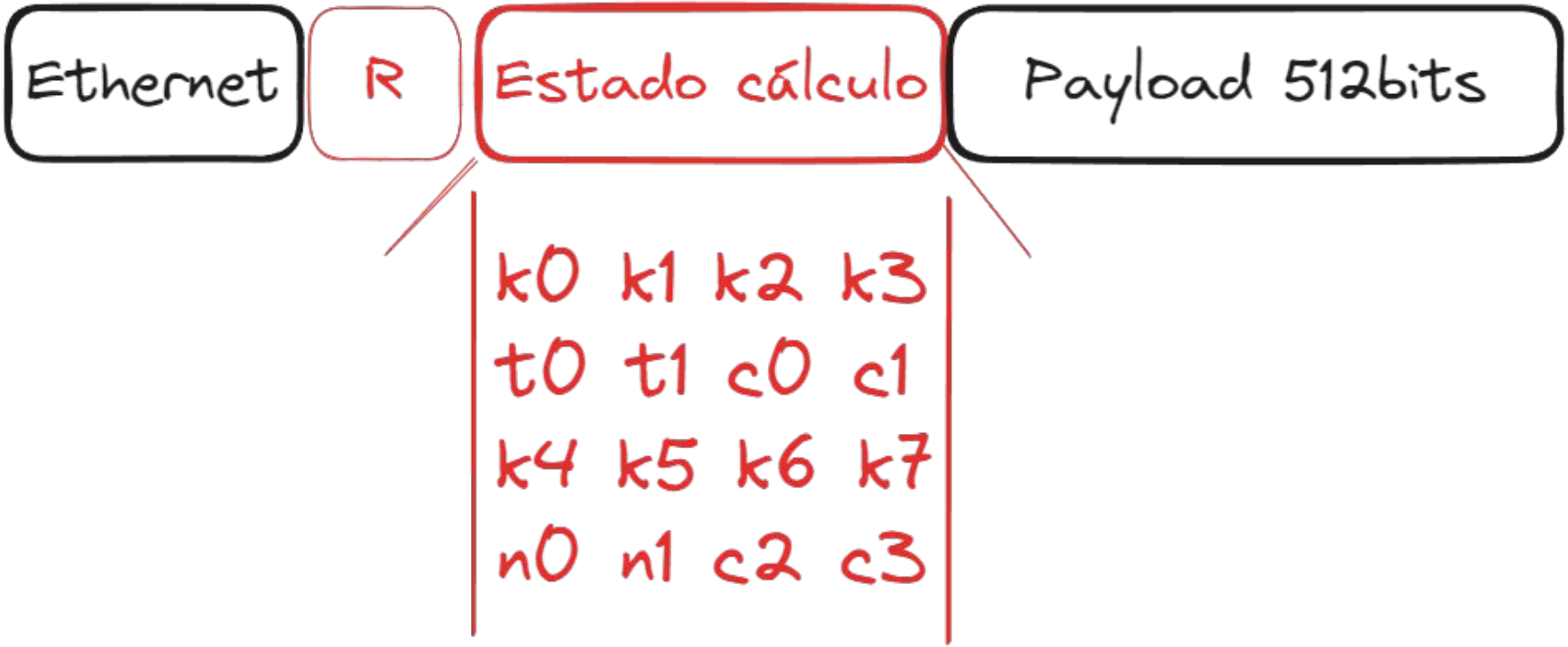


Vazão do Forro14 utilizando diferentes qtdes de portas para recirculação

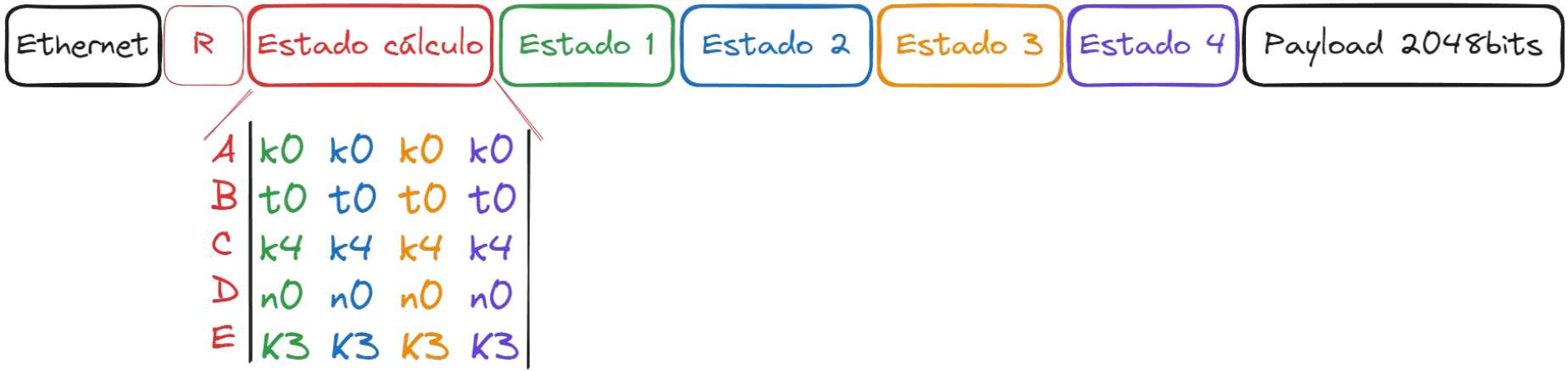
Backup


Ethernet
ChaCha_PreHeader
Modo + Rodada + Bloco
Estado cálculo
c0 c1 c2 c3
k0 k1 k2 k3
k4 k5 k6 k7
t0 t1 n0 n1
Payload 512 a 3072bits

Backup


Ethernet
R
Estado cálculo
Payload 512bits
k0 k1 k2 k3
t0 t1 c0 c1
k4 k5 k6 k7
n0 n1 c2 c3

Backup


Ethernet
R
Estado cálculo
Estado 1
Estado 2
Estado 3
Estado 4
Payload 2048bits
A k0 k0 k0 k0
B t0 t0 t0 t0
C k4 k4 k4 k4
D n0 n0 n0 n0
E K3 K3 K3 K3

Backup

Forro

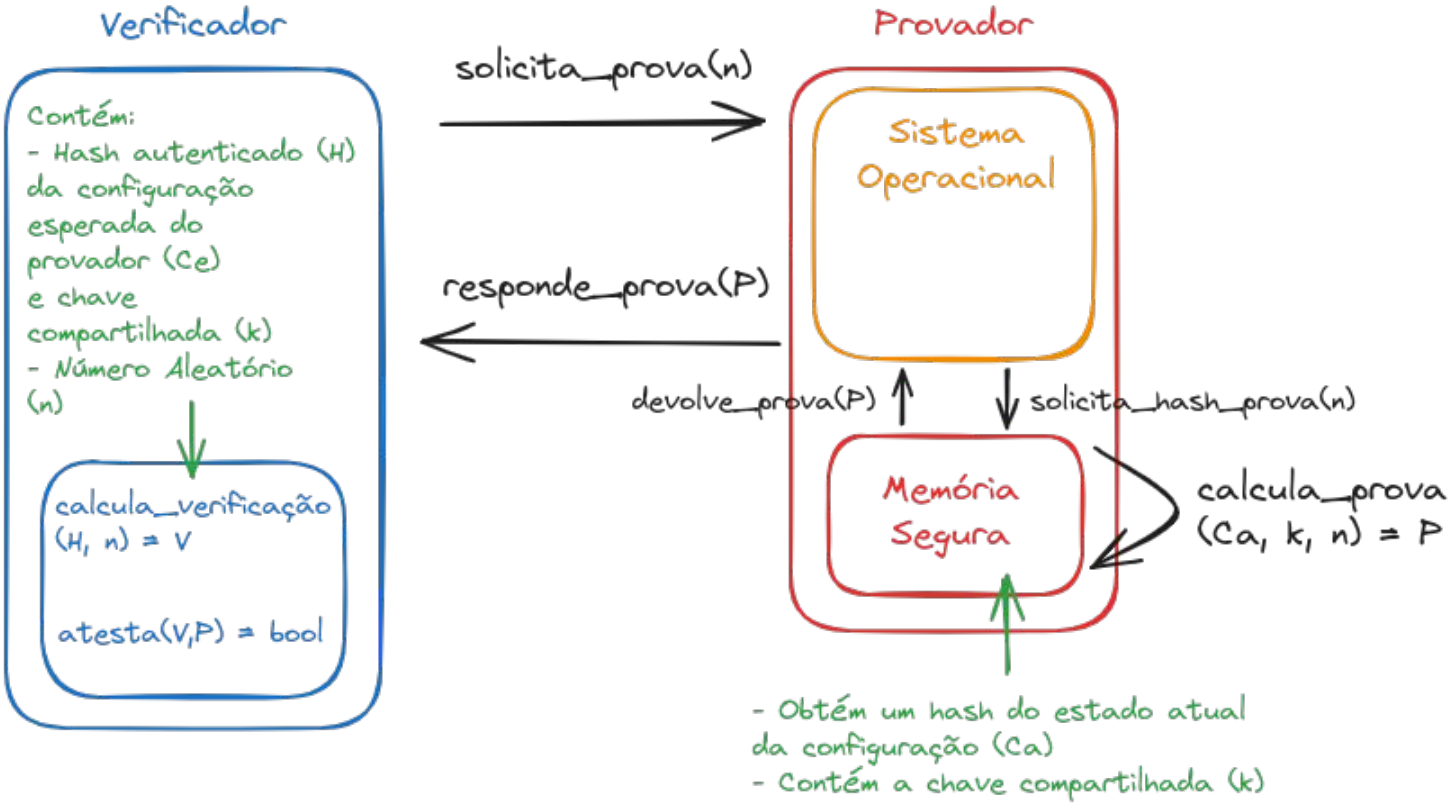
k0	k1	k2	k3
t0	t1	c0	c1
k4	k5	k6	k7
n0	n1	c2	c3

k = chave (256 bits)
t = contador (64 bits)
c = constantes (128 bits)
n = nonce (64 bits)

ChaCha

c0	c1	c2	c3
k0	k1	k2	k3
k4	k5	k6	k7
t0	t1	n0	n1

Backup


Verificador
Contém:
- Hash autenticado (H) da configuração esperada do provador (Ce) e chave compartilhada (k)
- Número Aleatório (n)
calcula_verificação (H, n) = V
atesta(V,P) = bool
solicita_prova(n)
responde_prova(P)
Provador
Sistema Operacional
devolve_prova(P)
solicita_hash_prova(n)
Memória Segura
calcula_prova (Ca, k, n) = P
- Obtém um hash do estado atual da configuração (Ca)
- Contém a chave compartilhada (k)

Backup

```
action i0_cipher () {
    hdr.forro_payload.linha0 = hdr.forro_payload.linha0 ^ hdr.forro_cipher.linha0;
    hdr.forro_payload.linha1 = hdr.forro_payload.linha1 ^ hdr.forro_cipher.linha1;
    hdr.forro_payload.linha2 = hdr.forro_payload.linha2 ^ hdr.forro_cipher.linha2;
    hdr.forro_payload.linha3 = hdr.forro_payload.linha3 ^ hdr.forro_cipher.linha3;

    // Definindo porta de saida e pulando Egress
    ig_tm_md.ucast_egress_port = 0x1;
    ig_tm_md.bypass_egress = 0x1;

    // Limpando cabeçalhos de rodada e estado para saida
    hdr.forro_rodada.setInvalid();
    hdr.forro_cipher.setInvalid();
    exit;
}
```

Backup

```
action i0_init(
    hashword_t key0, hashword_t key1, hashword_t key2, hashword_t key3,
    hashword_t key4, hashword_t key5, hashword_t key6, hashword_t key7,
    hashword_t n0_k3, hashword_t n1)
    {
    hdr.forro_rodada.setValid();
    hdr.forro_state.setValid();

    hdr.forro_state.v0 = key0;
    hdr.forro_state.v1 = key1;
    hdr.forro_state.v2 = key2;
    hdr.forro_state.v3 = key3;
    hdr.forro_state.v4 = 0x0;
    hdr.forro_state.v5 = 0x0;
    hdr.forro_state.v6 = H0;
    hdr.forro_state.v7 = H1;
    hdr.forro_state.v8 = key4;
    hdr.forro_state.v9 = key5;
    hdr.forro_state.v10 = key6;
    hdr.forro_state.v11 = key7;
    hdr.forro_state.v12 = n0_k3; // adiantando passo 1 do qr0
    hdr.forro_state.v13 = n1;
    hdr.forro_state.v14 = H2;
    hdr.forro_state.v15 = H3;

    //incrementando uma rodada
    hdr.forro_rodada.rodada = 1;

    //Alterando etheretype para o tipo do Forro14
    //  hdr.ethernet.ether_type = 0xABCD;
}
```

Backup

```
action i0_init(
    hashword_t key0, hashword_t key1, hashword_t key2, hashword_t key3,
    hashword_t key4, hashword_t key5, hashword_t key6, hashword_t key7,
    hashword_t n0_k3, hashword_t n1)
    {
    hdr.forro_rodada.setValid();
    hdr.forro_state.setValid();

    hdr.forro_state.v0 = key0;
    hdr.forro_state.v1 = key1;
    hdr.forro_state.v2 = key2;
    hdr.forro_state.v3 = key3;
    hdr.forro_state.v4 = 0x0;
    hdr.forro_state.v5 = 0x0;
    hdr.forro_state.v6 = H0;
    hdr.forro_state.v7 = H1;
    hdr.forro_state.v8 = key4;
    hdr.forro_state.v9 = key5;
    hdr.forro_state.v10 = key6;
    hdr.forro_state.v11 = key7;
    hdr.forro_state.v12 = n0_k3; // adiantando passo 1 do qr0
    hdr.forro_state.v13 = n1;
    hdr.forro_state.v14 = H2;
    hdr.forro_state.v15 = H3;

    //incrementando uma rodada
    hdr.forro_rodada.rodada = 1;

    //Alterando etheretype para o tipo do Forro14
    //  hdr.ethernet.ether_type = 0xABCD;
}
```

Backup

```
action e11_qr7_fin(in hashword_t key0, in hashword_t key1, in hashword_t key2, in hashword_t key3,
    in hashword_t key4, in hashword_t key5, in hashword_t key6, in hashword_t key7,
    in hashword_t n0, in hashword_t n1)
{
    hdr.forro_state.v0 = hdr.forro_state.v0 + key0;
    hdr.forro_state.v1 = hdr.forro_state.v1 + key1;
    hdr.forro_state.v2 = hdr.forro_state.v2 + key2;
    hdr.forro_state.v3 = copy32_0.get(hdr.forro_state.v3[23:0] ++ hdr.forro_state.v3[31:24]) + meta.key3; //Shift + Add
    hdr.forro_state.v4 = hdr.forro_state.v4 + 0x0;
    hdr.forro_state.v5 = hdr.forro_state.v5 + 0x0;
    hdr.forro_state.v6 = hdr.forro_state.v6 + C0;
    hdr.forro_state.v7 = hdr.forro_state.v7 + C1;
    hdr.forro_state.v8 = hdr.forro_state.v8 + key4;
    hdr.forro_state.v9 = hdr.forro_state.v9 + key5;
    hdr.forro_state.v10 = hdr.forro_state.v10 + key6;
    hdr.forro_state.v11 = hdr.forro_state.v11 + key7;
    hdr.forro_state.v12 = hdr.forro_state.v12 + n0;
    hdr.forro_state.v13 = hdr.forro_state.v13 + n1;
    hdr.forro_state.v14 = hdr.forro_state.v14 + C2;
    hdr.forro_state.v15 = hdr.forro_state.v15 + C3;

    hdr.forro_rodada.rodada = hdr.forro_rodada.rodada + 1;
}
```

Backup

```
Hash<bit<32>>(HashAlgorithm_t.IDENTITY) copy32_0;
```

```
hdr.forro_state.v3 = copy32_0.get(hdr.forro_state.v3[23:0] ++ hdr.forro_state.v3[31:24]) + meta.key3; //Shift + Add
```

Backup

```
header forro_state_t {
    hashword_t  v0; //k0
    hashword_t  v1; //k1
    hashword_t  v2; //k2
    hashword_t  v3; //k3
    hashword_t  v4; //t0
    hashword_t  v5; //t1
    hashword_t  v6; //c0
    hashword_t  v7; //c1
    hashword_t  v8; //k4
    hashword_t  v9; //k5
    hashword_t  v10; //k6
    hashword_t  v11; //k7
    hashword_t  v12; //v0
    hashword_t  v13; //v1
    hashword_t  v14; //c2
    hashword_t  v15; //c3
}
```

```
header forro_state_cipher_t {
    bit<128> linha0;
    bit<128> linha1;
    bit<128> linha2;
    bit<128> linha3;
}

header forro_payload_t {
    bit<128> linha0;
    bit<128> linha1;
    bit<128> linha2;
    bit<128> linha3;
}
```

Backup

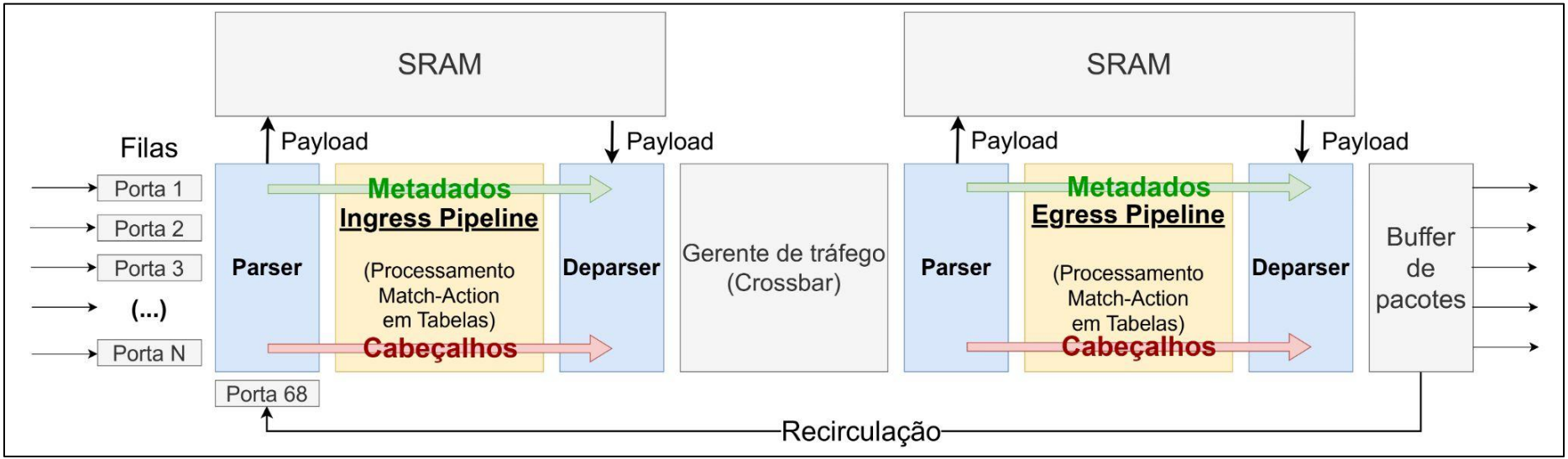
```
p4.Ingress.tbl_forro_ig0.add_with_i0_init(rodada=0)

p4.Ingress.tbl_forro_ig0.add_with_i0_qr0(rodada=1)
p4.Ingress.tbl_forro_ig1.add_with_i1_qr0(rodada=1)
p4.Ingress.tbl_forro_ig2.add_with_i2_qr0(rodada=1)
p4.Ingress.tbl_forro_ig3.add_with_i3_qr0(rodada=1)
p4.Ingress.tbl_forro_ig4.add_with_i4_qr0(rodada=1)
p4.Ingress.tbl_forro_ig5.add_with_i5_qr0(rodada=1)
p4.Ingress.tbl_forro_ig6.add_with_i6_qr0(rodada=1)
p4.Ingress.tbl_forro_ig7.add_with_i7_qr0(rodada=1)
p4.Ingress.tbl_forro_ig8.add_with_i8_qr0(rodada=1)
p4.Ingress.tbl_forro_ig9.add_with_i9_qr0(rodada=1)
p4.Ingress.tbl_forro_ig10.add_with_i10_qr0(rodada=1
p4.Ingress.tbl_forro_ig11.add_with_i11_qr0(rodada=1
p4.Egress.tbl_forro_eg0.add_with_e0_qr1(rodada=2)
p4.Egress.tbl_forro_eg1.add_with_e1_qr1(rodada=2)
p4.Egress.tbl_forro_eg2.add_with_e2_qr1(rodada=2)
p4.Egress.tbl_forro_eg3.add_with_e3_qr1(rodada=2)
p4.Egress.tbl_forro_eg4.add_with_e4_qr1(rodada=2)
p4.Egress.tbl_forro_eg5.add_with_e5_qr1(rodada=2)
p4.Egress.tbl_forro_eg6.add_with_e6_qr1(rodada=2)
p4.Egress.tbl_forro_eg7.add_with_e7_qr1(rodada=2)
p4.Egress.tbl_forro_eg8.add_with_e8_qr1(rodada=2)
p4.Egress.tbl_forro_eg9.add_with_e9_qr1(rodada=2)
p4.Egress.tbl_forro_eg10.add_with_e10_qr1(rodada=2)
p4.Egress.tbl_forro_eg11.add_with_e11_qr1(rodada=2)
```

Backup

```
private bool IsEven(int number){
    if (number == 1) return false;
    else if (number == 2) return true;
    else if (number == 3) return false;
    else if (number == 4) return true;
    else if (number == 5) return false;
    else if (number == 6) return true;
    else if (number == 7) return false;
    else if (number == 8) return true;
    else if (number == 9) return false;
    else if (number == 10) return true;
    else if (number == 11) return false;
    else if (number == 12) return true;
    else if (number == 13) return false;
    else if (number == 14) return true;
    else if (number == 15) return false;
    else if (number == 16) return true;
    else if (number == 17) return false;
    else if (number == 18) return true;
    else if (number == 19) return false;
    else if (number == 20) return true;
    else if (number == 21) return false;
    else if (number == 22) return true;
```

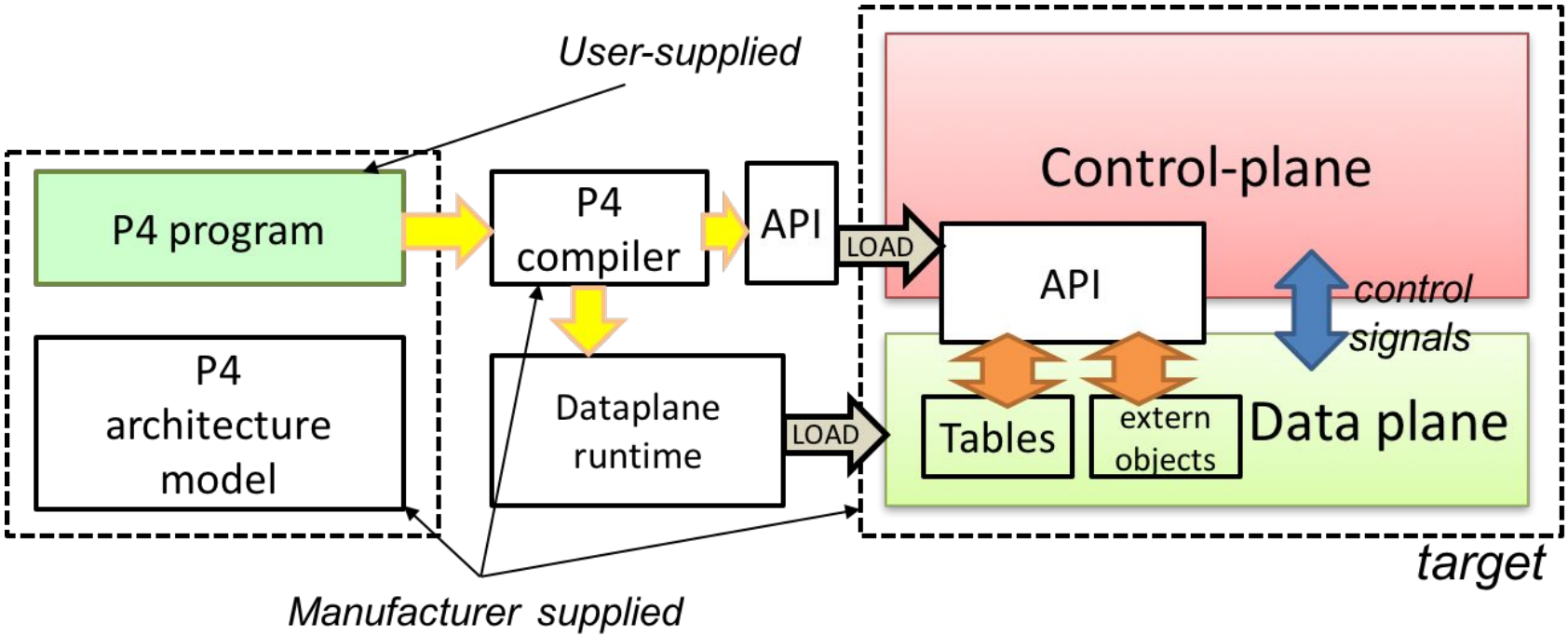
Backup



Arquitetura referência de um switch programável com P4

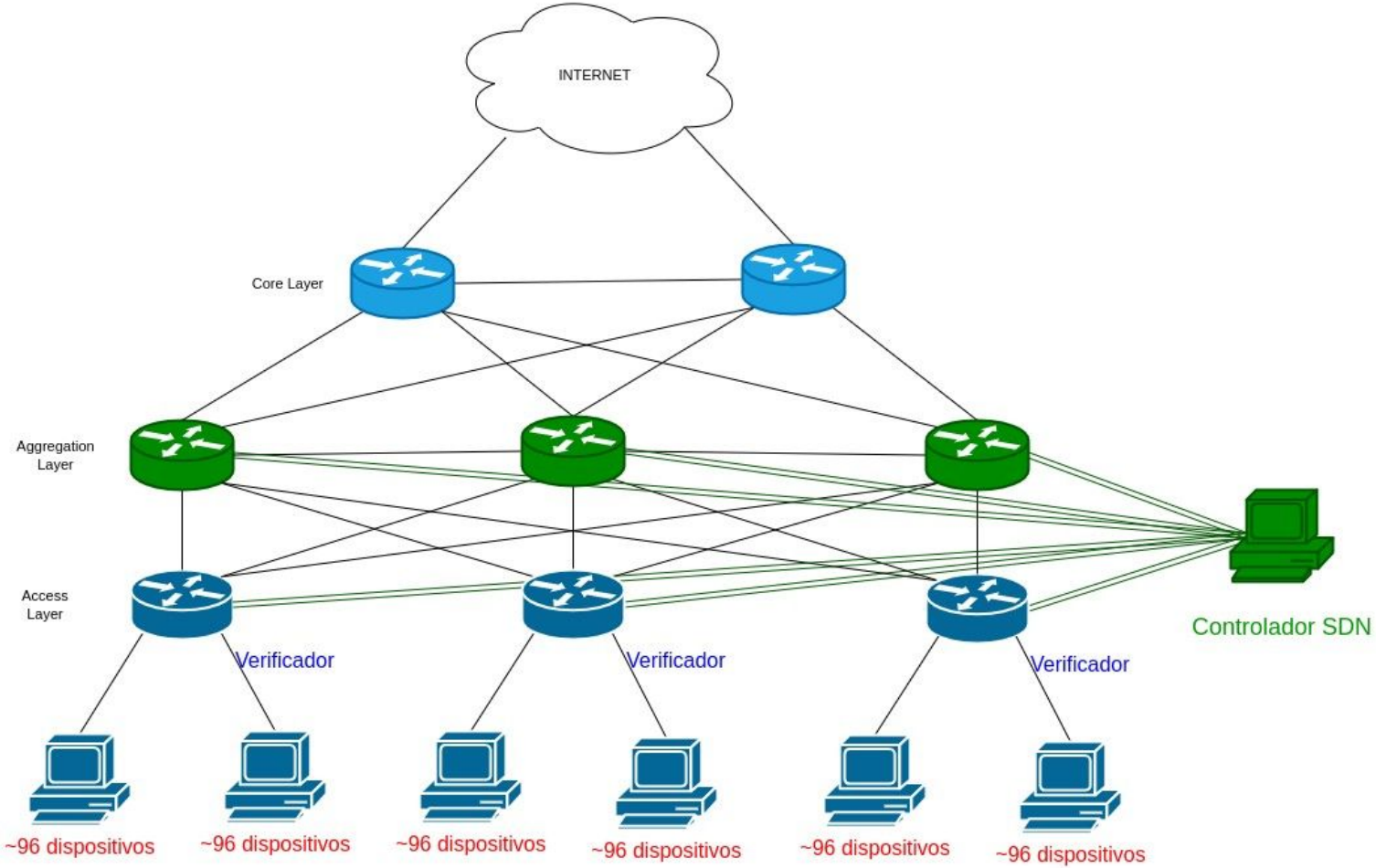
Fonte da imagem: Adaptada de *Peterson, L., Cascone, C., and Davie, B. (2021). Software-Defined Networks: A Systems Approach.*

Backup

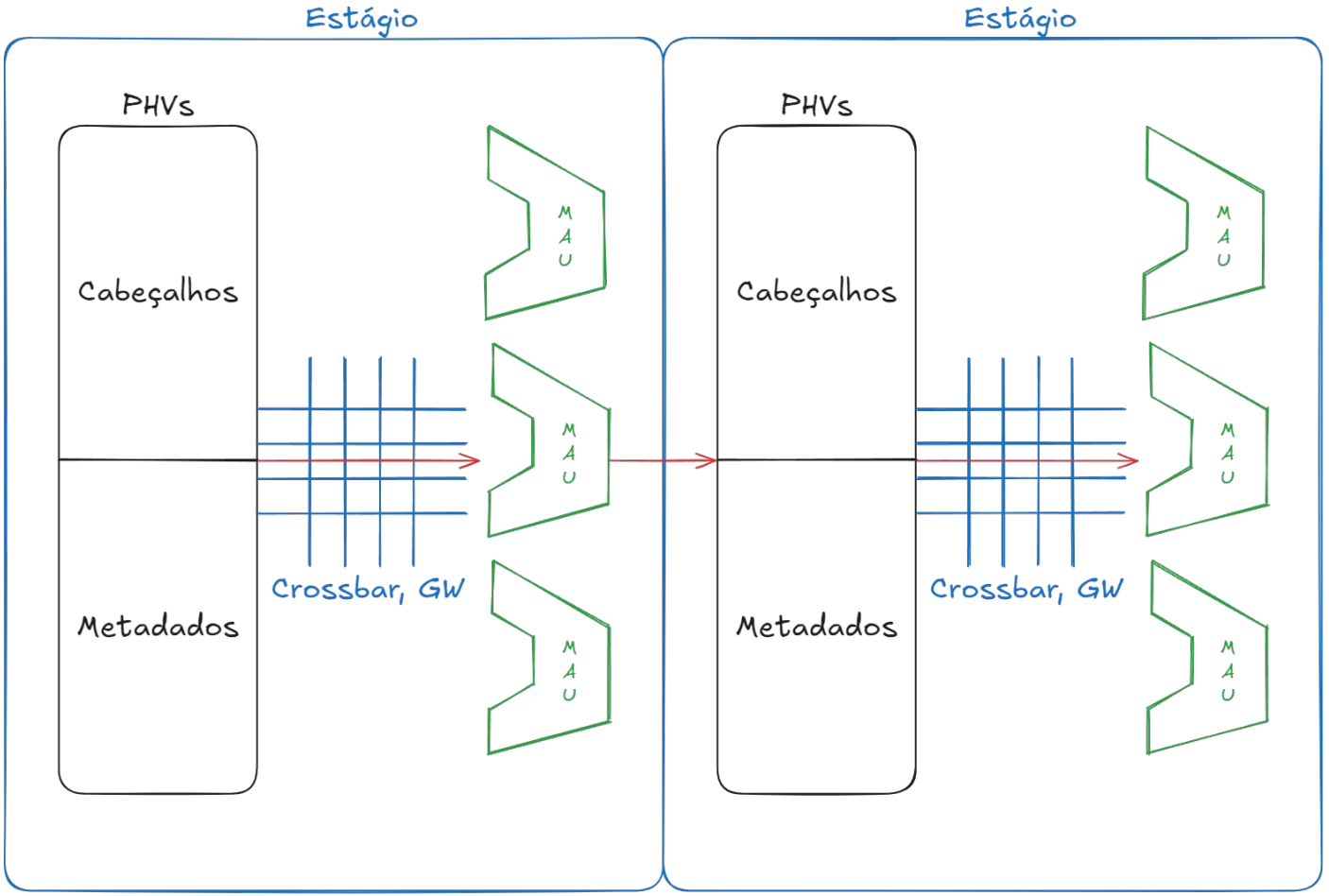


Fonte: https://p4.org/p4-spec/docs/P4-16-v1.0.0-spec.html

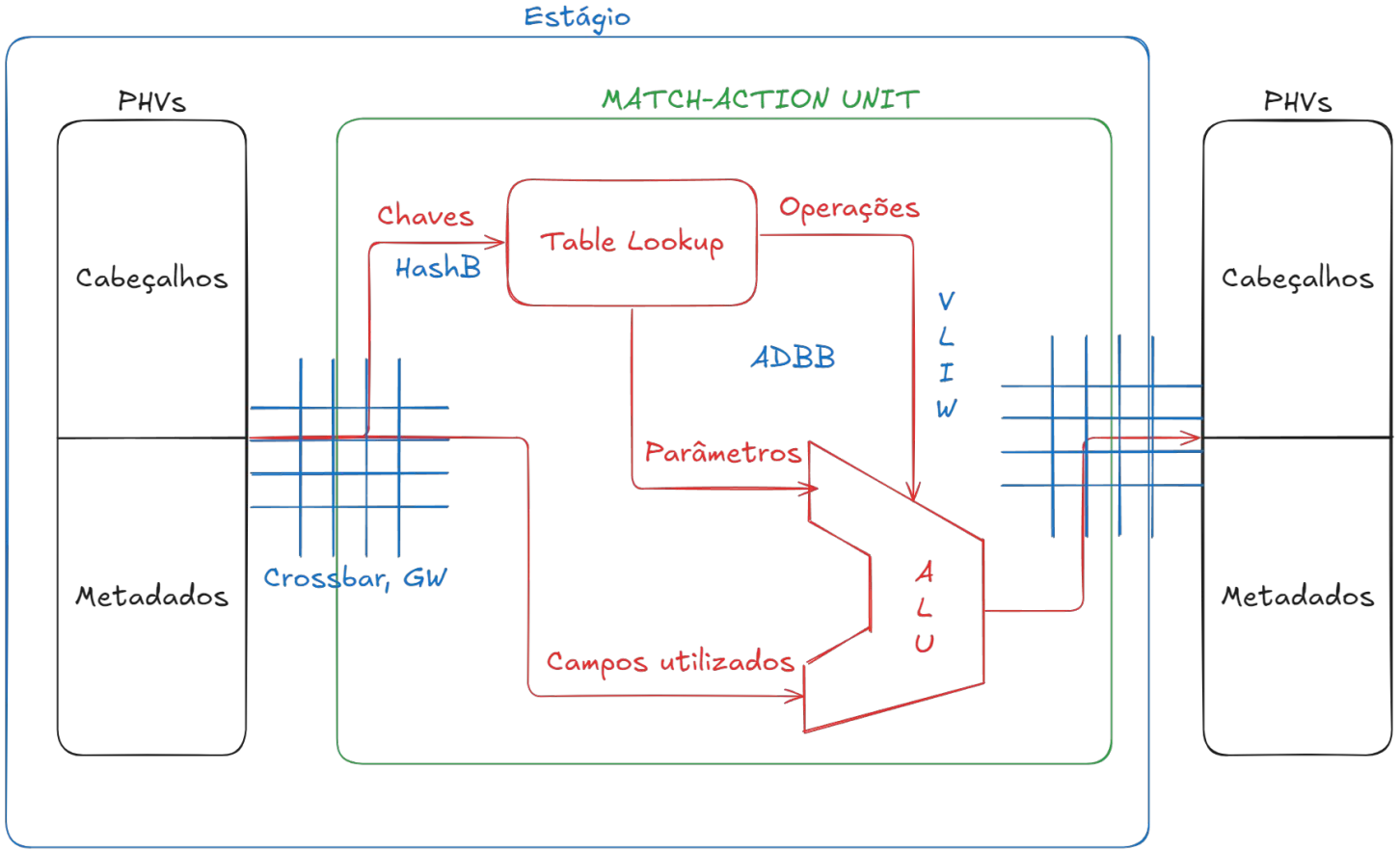
Backup


INTERNET
Core Layer
Aggregation Layer
Access Layer
Verificador
Verificador
Verificador
Controlador SDN
~96 dispositivos
~96 dispositivos
~96 dispositivos
~96 dispositivos
~96 dispositivos
~96 dispositivos

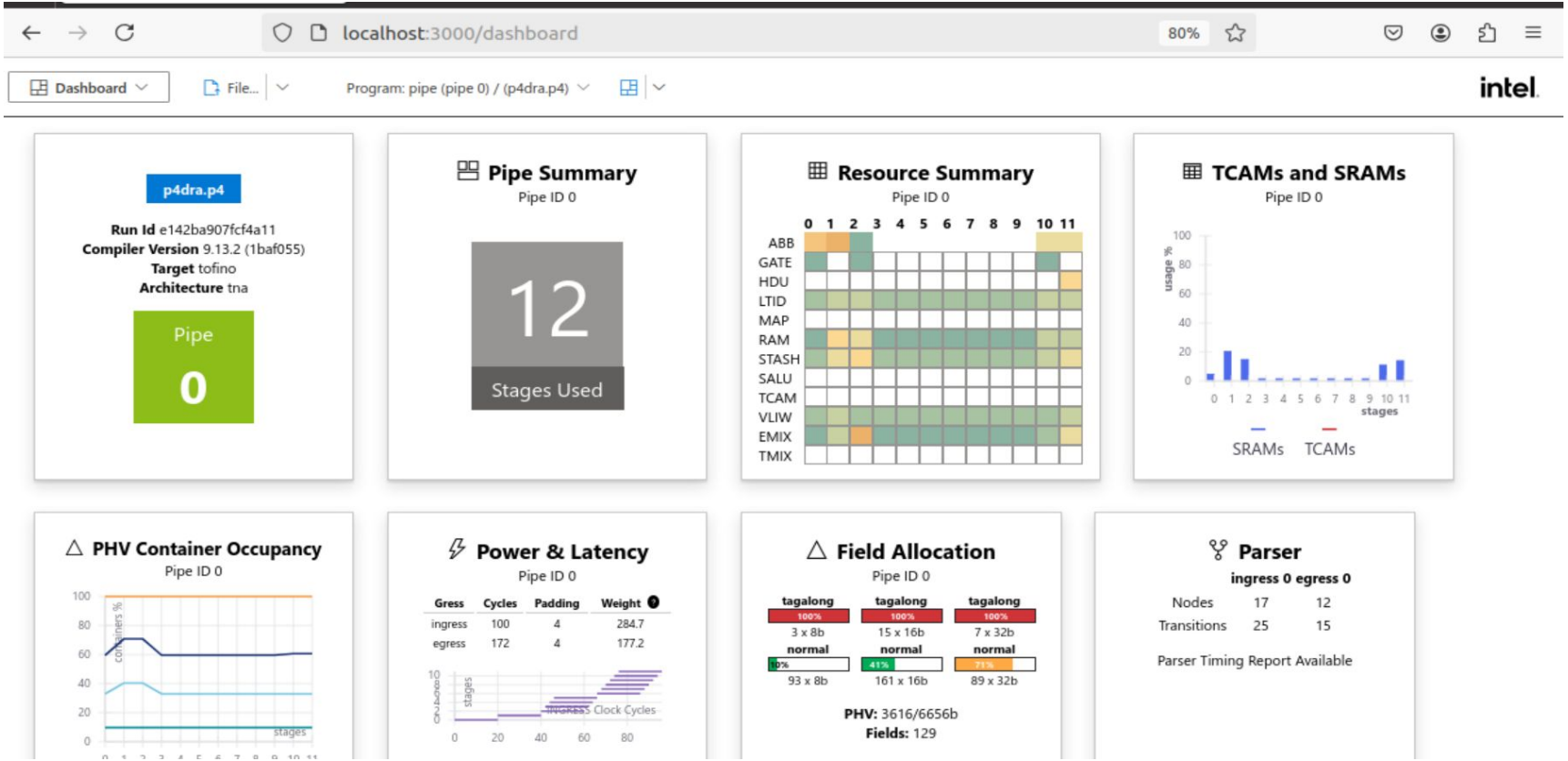
Backup


Estágio
Estágio
PHVs
Cabeçalhos
Metadados
MAU
MAU
MAU
Crossbar, GW
PHVs
Cabeçalhos
Metadados
MAU
MAU
MAU
Crossbar, GW

Backup


Estágio
PHVs
MATCH-ACTION UNIT
PHVs
Cabeçalhos
Chaves
HashB
Table Lookup
Operações
VLIW
ADBB
Cabeçalhos
Parâmetros
Metadados
Crossbar, GW
ALU
Campos utilizados
Metadados

Backup


localhost:3000/dashboard
80%
Dashboard
File...
Program: pipe (pipe 0) / (p4dra.p4)
intel
p4dra.p4
Run Id e142ba907fcf4a11
Compiler Version 9.13.2 (1baf055)
Target tofino
Architecture tna
Pipe
0
Pipe Summary
Pipe ID 0
12
Stages Used
Resource Summary
Pipe ID 0
0 1 2 3 4 5 6 7 8 9 10 11
ABB
GATE
HDU
LTID
MAP
RAM
STASH
SALU
TCAM
VLIW
EMIX
TMIX
TCAMs and SRAMs
Pipe ID 0
usage %
stages
SRAMs
TCAMs
PHV Container Occupancy
Pipe ID 0
containers %
stages
Power & Latency
Pipe ID 0
Gress Cycles Padding Weight
ingress 100 4 284.7
egress 172 4 177.2
stages
INGRESS Clock Cycles
Field Allocation
Pipe ID 0
tagalong tagalong tagalong
100% 100% 100%
3 x 8b 15 x 16b 7 x 32b
normal normal normal
0% 41% 71%
93 x 8b 161 x 16b 89 x 32b
PHV: 3616/6656b
Fields: 129
Parser
ingress 0 egress 0
Nodes 17 12
Transitions 25 15
Parser Timing Report Available