Consenso Bizantino Baseado em uma Camada de Rede com Ordenação de Mensagens Tolerante a Intrusões

Gabriel Faustino[1] Eduardo Alchieri[1] Giovanni Venâncio[2]

Vinicius Fulber-Garcia[2] Elias P. Duarte Jr.[2]

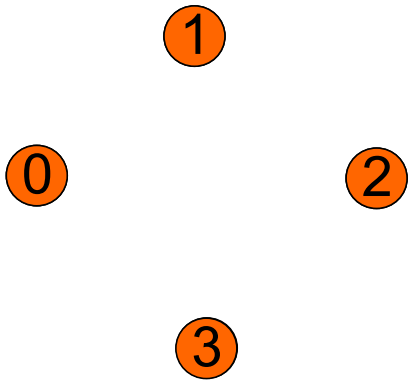
[1]Departamento de Ciência da Computação, Universidade de Brasília
[2]Departamento de Informática, Universidade do Paraná

Tópicos da Apresentação

- Introdução
- NOPaxos e NeoBFT
- NsoBFT e variantes
- Experimentos
- Conclusões

Introdução

- O consenso e, de modo geral, os algoritmos de acordo formam a base para a solução de diversos problemas relacionados ao desenvolvimento de sistemas distribuídos confiáveis:
  - Replicação máquina de estados: Azure, Zookeeper, etc.
  - Blockchains



- Propriedades:
  - Validade
  - Acordo
  - Terminação

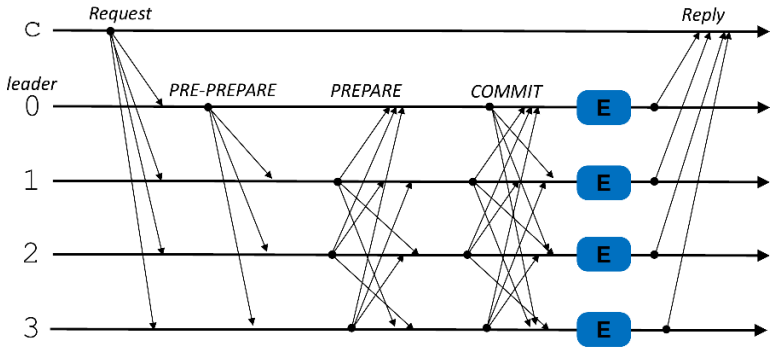
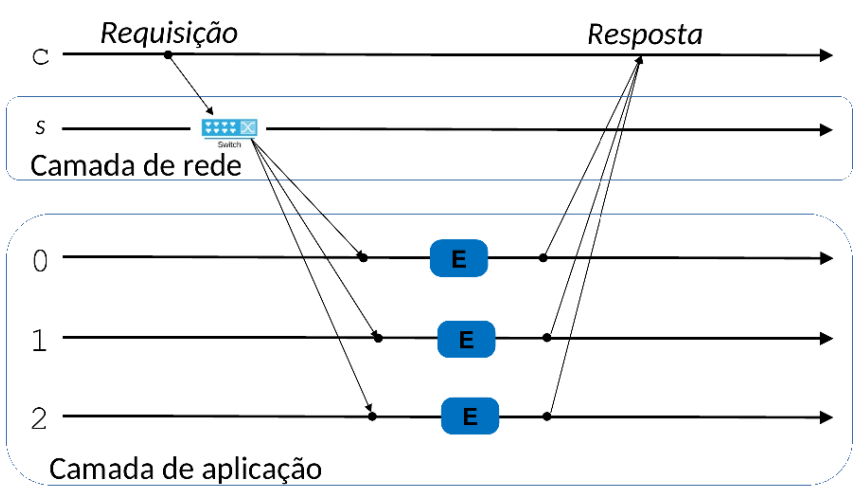


Figure 4.6. PBFT normal case protocol.

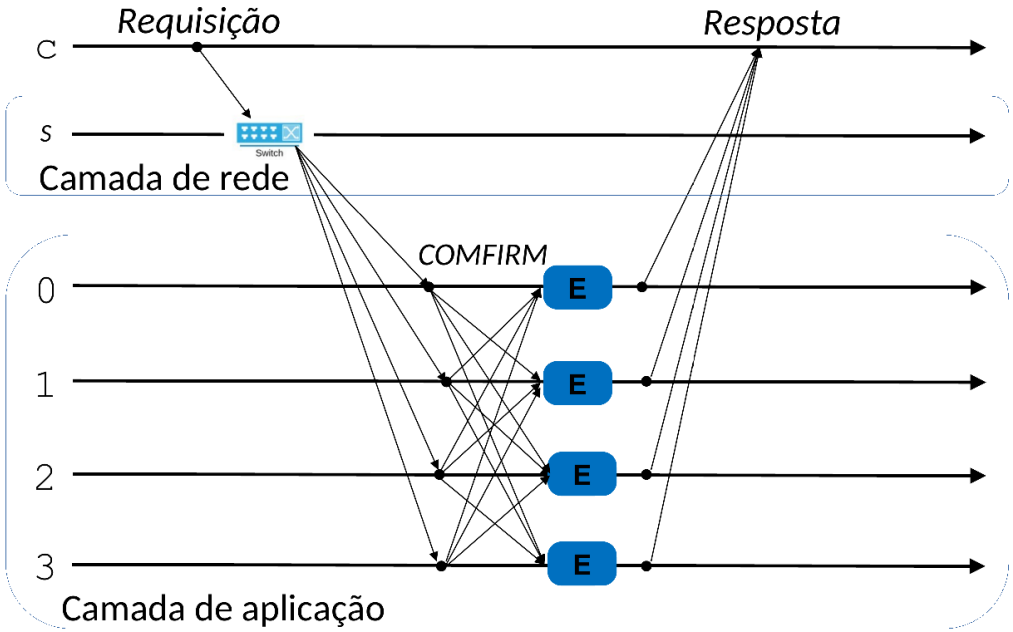
Introdução

- O Paxos é um protocolo de consenso clássico
  - PBFT foi o primeiro protocolo de consenso BFT a considerar aspectos práticos
  - Implementam todas as propriedades na camada de aplicação
- Estudos recentes mostram que dividir as responsabilidades sobre as propriedades entre as camadas de rede e de aplicação melhora o desempenho do sistema
  - NOPaxos: utiliza um sequenciador na camada de rede para ordenação (acordo)
  - NeoBFT: utiliza um sequenciador que utiliza assinaturas para prover um serviço autenticado

NOPaxos e NeoBFT



(a) NOPaxos.



(b) NeoBFT.

NOPaxos e NeoBFT

- Como implementar o sequenciador na camada de rede?
  - NOPaxos:
    - In-switch: switches (P4)
    - Hardware Middlebox: protótipo usando um middlebox (processador de rede Cavium Octeon II CN68XX) junto ao switch.
    - End-host: servidor convencional
  - NeoBFT:
    - In-switch HMAC (HalfSipHash): vetor de HMAC
    - FPGA: coprocessador criptográfico para gerar assinaturas

NsoBFT

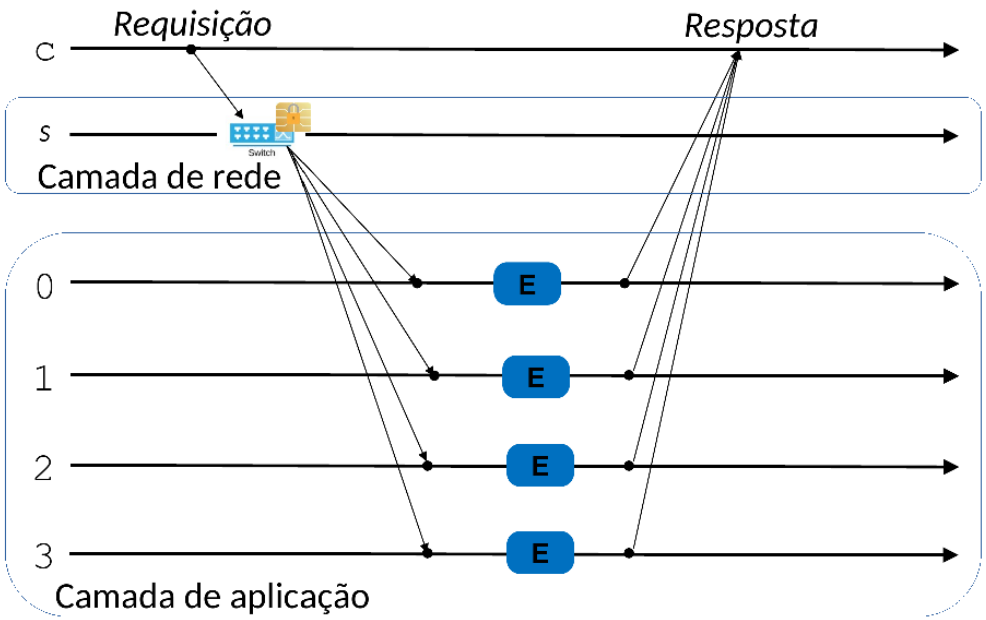
- Propomos o NsoBFT (Network secure ordered BFT)
  - Utiliza um componente seguro para implementar um sequenciador autenticado e tolerante a intrusões
  - USIG - Gerador de Identificadores Sequenciais Únicos
    - Impede que um sequenciador malicioso envie mensagens diferentes com um mesmo número de sequência ou ainda atribua diferentes números de sequência para uma mesma mensagem
    - Duas operações:
      - UI = createUI(m) e verifyUI(PK,UI,m)
    - Pode ser implementado usando apenas hashes ou assinaturas digitais
    - Pode ser implementado com diferentes níveis de isolamento: máquinas virtuais ou até mesmo módulos seguros de hardware.

NsoBFT

- Modelo de sistema
  - O sistema é formado por um conjunto ilimitado de clientes e *n* réplicas
  - Existe pelo menos um sequenciador correto instalado na camada de rede
    - Utiliza o USIG para gerar números de sequência
  - Assume o modelo de falhas bizantinas
  - No máximo *f* réplicas faltosas de $n \geq 3f+1$
  - Sistema parcialmente síncrono para garantir terminação

NsoBFT

- Execução normal



1. O cliente realiza um multicast da requisição r
2. O sequenciador utiliza o USIG para adicionar um contador em r
3. As réplicas recebem r e verificam se o contador é válido
   a) Verificam se r é a próxima requisição a ser executada
   b) Executam r, inserem r em um log, enviam a resposta ao cliente
4. O cliente aguarda por 2f+1 respostas iguais e completa a operação

NsoBFT

- Troca de visão
  - Quando o sequenciador é suspeito de ser malicioso, é necessário escolher outro
  - É necessário determinar até qual ponto no log as requisições devem ser preservadas
    - Todas as requisições que foram respondidas por pelo menos *2f+1* réplicas aos clientes devem permanecer no log

NsoBFT

- Protocolo de troca de visão
  - Quando uma réplica suspeita que o sequenciador está faltoso, ela envia uma mensagem de VIEW-CHANGE para as outras réplicas
    - Quando uma réplica recebe uma mensagem de VIEW-CHANGE, ela retransmite essa mensagem para as demais réplicas
  - Quando uma réplica recebe *f+1* mensagens de VIEW-CHANGE, ela também envia uma mensagem de VIEW-CHANGE
    - Inicia um protocolo de consenso para determinar até qual ponto do log deve ser mantido
      - Uma réplica líder propõe o seu log
      - As outras réplicas só aceitam uma proposta caso o seu log não esteja à frente do log proposto
    - Ao final do consenso, réplicas atualizam o log (executando requisições ou fazendo rollback)

MinNsoBFT e MinZyzNsoBFT

- Ambas reduzem a quantidade de réplicas para apenas *2f+1*
- O aspecto fundamental a ser garantido é que requisições já completas não sejam removidas do log das réplicas durante uma troca de visão

- MinNsoBFT: inspirada no MinBFT
  - Demanda um passo adicional de comunicação e utiliza quóruns de *f+1* réplicas

- MinZyzNsoBFT: inspirada no MinZyzzyva
  - Os clientes aguardam por todas as *2f+1* respostas para completar uma requisição

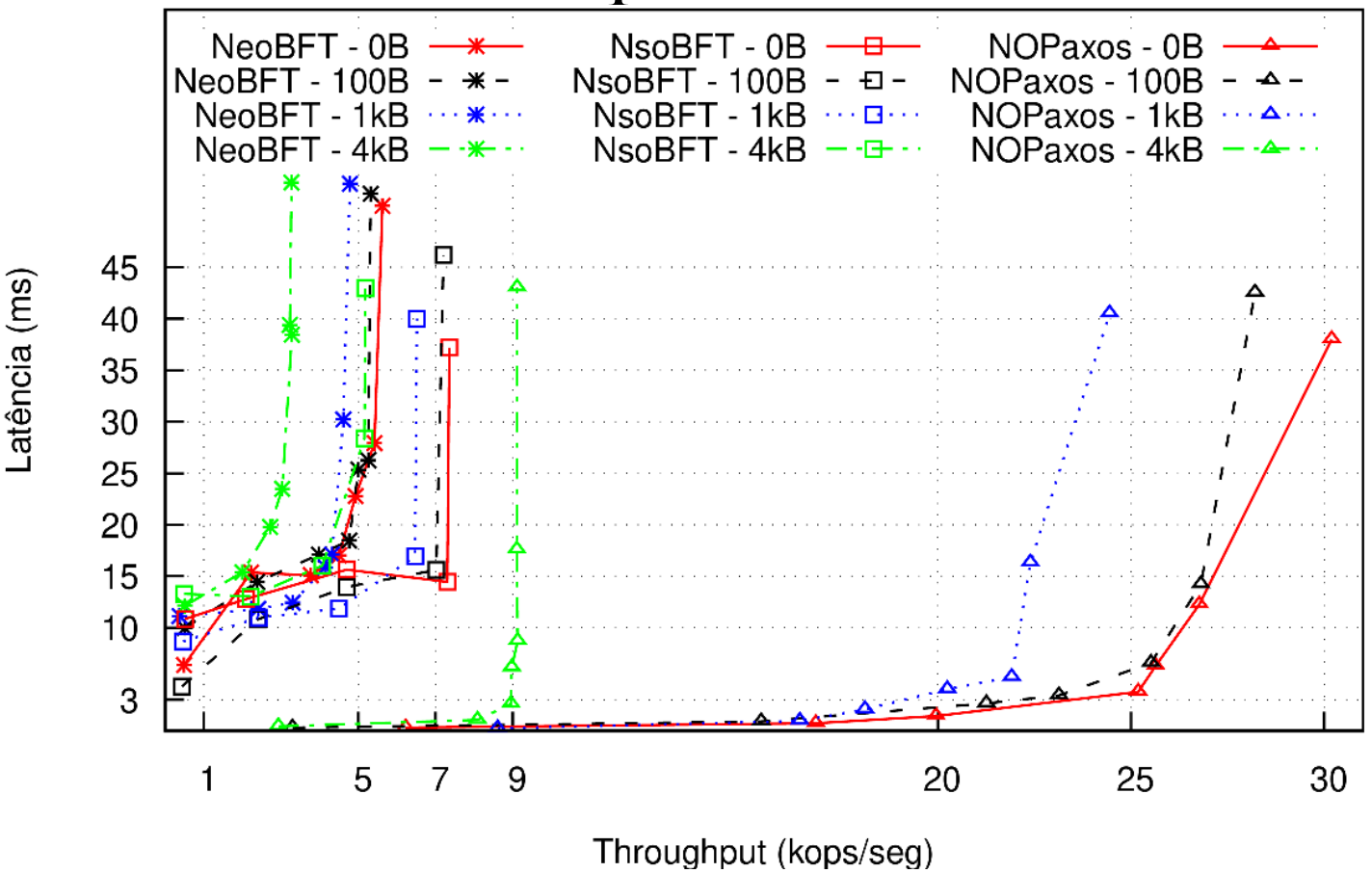
Comparação dos protocolos

Característica	Paxos	PBFT	NOPaxos	NeoBFT	**NsoBFT**	**MinNsoBFT**	**MinZyzNsoBFT**
Modelo de falhas	*crash*	bizantina	*crash*	bizantina	**bizantina**	**bizantina**	**bizantina**
Número de réplicas[1]	*2f+1*	*3f+1*	*2f+1*	*3f+1*	***3f+1***	***2f+1***	***2f+1***
Passos de comunicação	4	5	2	3	**2**	**3**	**2**
Complex. de comunicação[2]	$O(n^2)$	$O(n^2)$	$O(n)$	$O(n^2)$	**$O(n)$**	**$O(n^2)$**	**$O(n)$**
Componente seguro	-	-	-	-	**USIG**	**USIG**	**USIG**

Experimentos

- Os protocolos foram implementados em Java e testados no Emulab
  - 6 máquinas d430 (2.4 GHz E5-2630v3, com 8 núcleos e 2 threads por núcleo, 64GB de RAM e interface de rede gigabit) conectadas a um switch de 1Gb.
  - NOPaxos foi configurado com 3 réplicas
  - NeoBFT e o NsoBFT foram configuradas com 4 réplicas
  - Um sequenciador for executado como um end-host
  - Um número variado de clientes foram executados na máquina restante
- O serviço implementado é “vazio” (nada é processado nas réplicas)
- Assinaturas de 256 bits (*Bouncy Castle*)

Experimentos



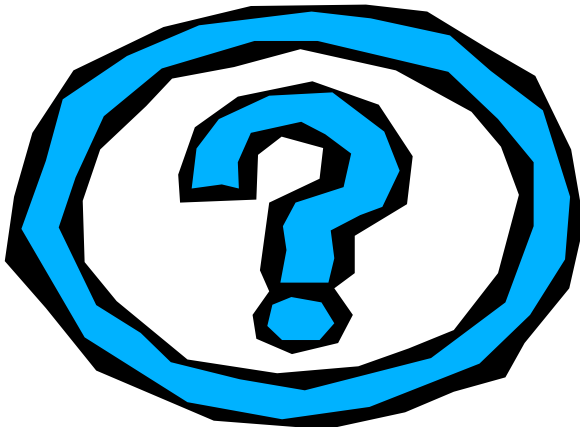
Conclusões

- Propomos um novo protocolo de consenso que apresenta avanços em relação ao NeoBFT
  - Reduz um passo de comunicação e torna o protocolo O(n)
- Existe um trade-off entre desempenho e resiliência

- Trabalhos futuros
  - Como implementar o sequenciador no switch?
    - Adicionar um coprocessador seguro
  - Implementar o sequenciador em uma NFV
    - Pode manter isolamento no sistema
  - Implementar e analisar o desempenho das variantes do NsoBFT

UnB | CIC

COMNET computer networks lab



Obrigado!