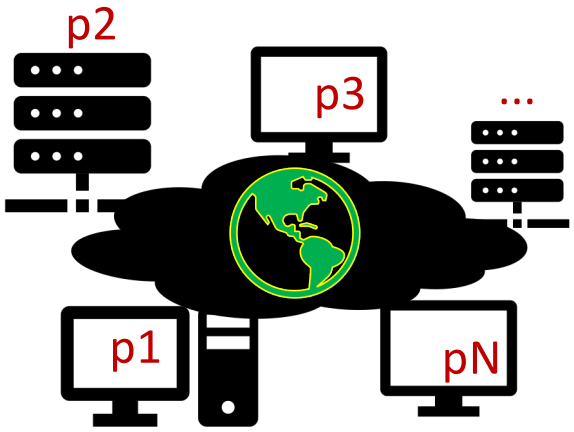
Fundamentos de Processamento Paralelo e Distribuído

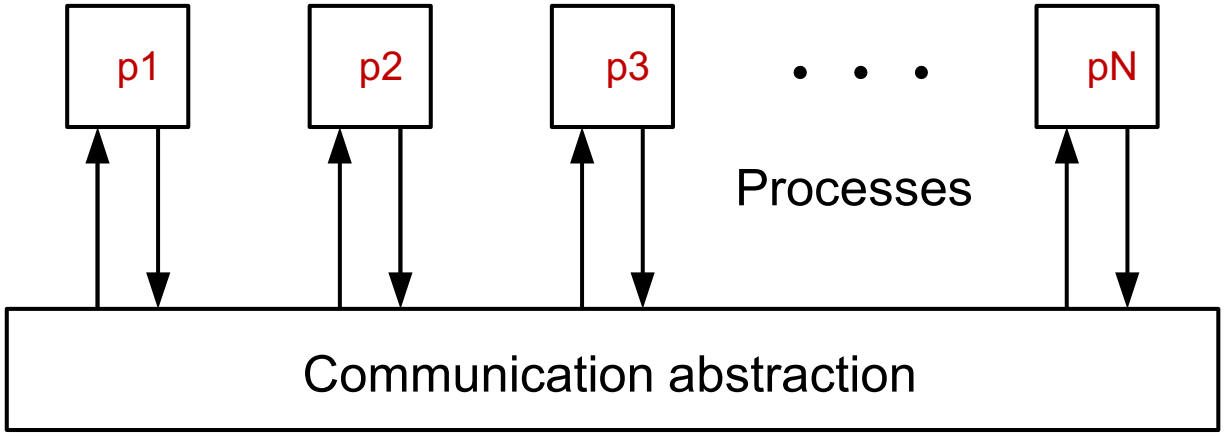
Processos distribuídos - Modelo construtivo

Fernando Luís Dotti – PUCRS

Material baseado no livro
Introduction to Reliable and Secure Distributed Programming
Christian Cachin, Rachid Gerraoui, Luís Rodrigues
e em material didático disponibilizado pelos autores

Abstrações para Programação Distribuída



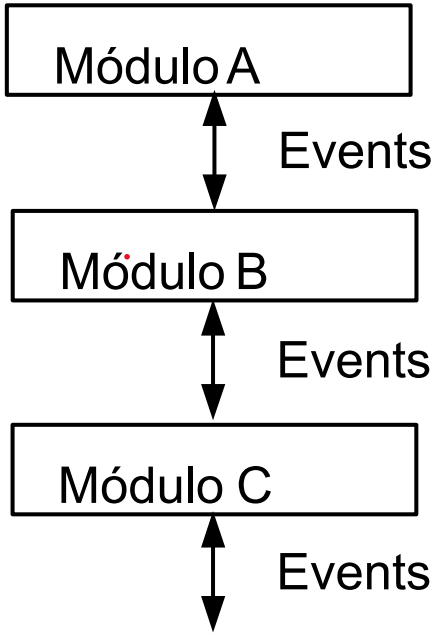


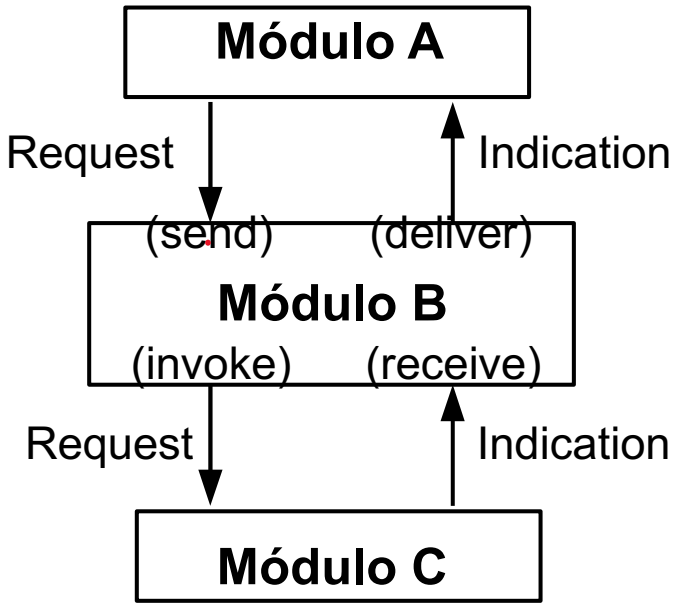
- Sistema com N processos $\Pi = \{p_1, \ldots p_N\}$
- Os processos "se conhecem"
- Se coordenam para uma tarefa comum

Material baseado no livro: *Introduction to Reliable and Secure Distributed Programming* *Christian Cachin, Rachid Gerraoui, Luís Rodrigues*

Processos

- Todo processo consiste em um conjunto de módulos
- Módulos se comunicam através de eventos





Cada módulo tem suas propriedades.

As propriedades devem ser satisfeitas com a funcionalidade do módulo inferior e seu algoritmo.

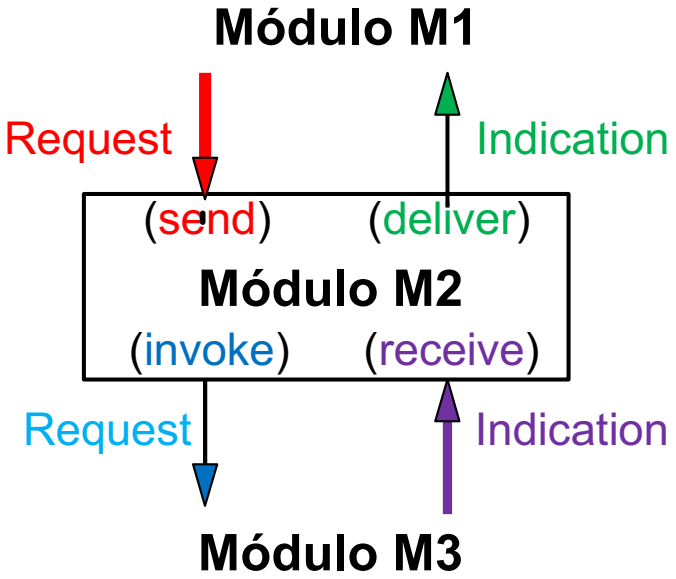
Argumentação

Material baseado no livro: *Introduction to Reliable and Secure Distributed Programming* *Christian Cachin, Rachid Gerraoui, Luís Rodrigues*

Módulos Reativos

- Modelo de especificação/ programação reativa

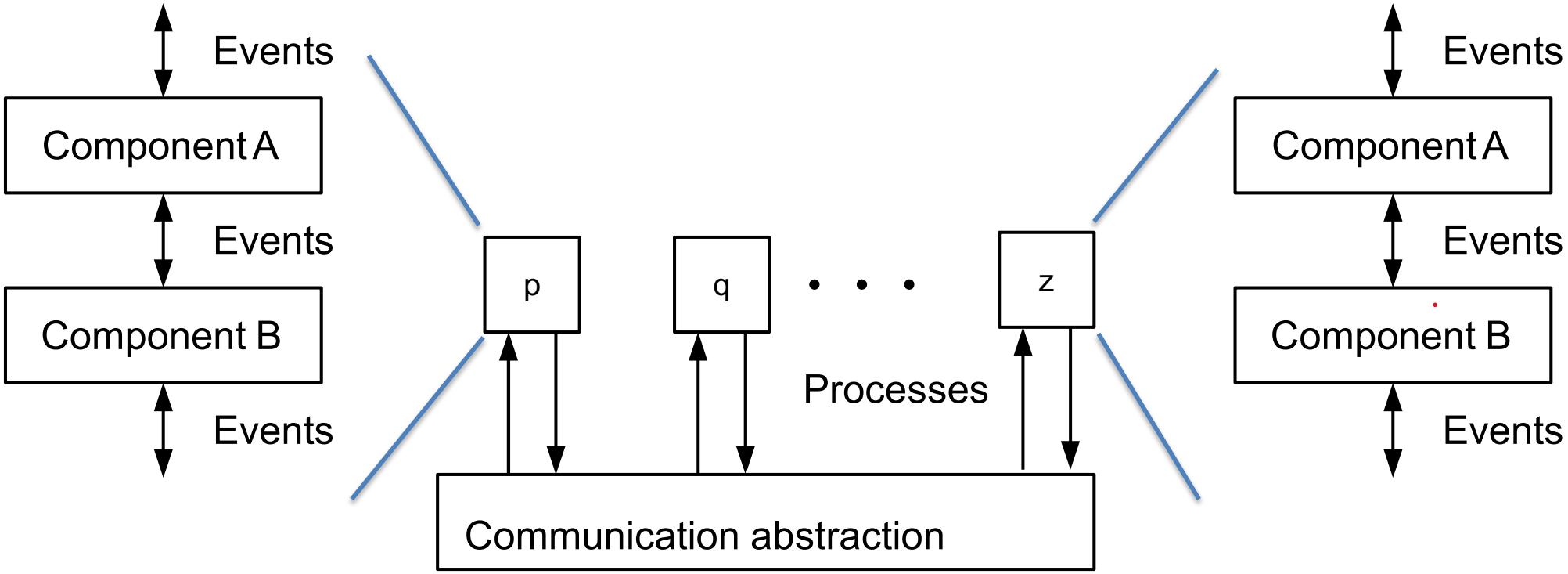
Upon event <M2, send |...> do
    trata pedido de envio;
    trigger <M3, invoke ...|...>;
Upon event <M2, receive |...> do
    trata pedido de envio;
    trigger <M3, indication ...|...>;



- Os módulos estão dispostos em camadas
- Eventos representam fluxo de comunicação ou controle
  - Os eventos de request fluem para baixo
  - Os eventos de indication fluem para cima

Material baseado no livro: *Introduction to Reliable and Secure Distributed Programming* *Christian Cachin, Rachid Gerraoui, Luís Rodrigues*

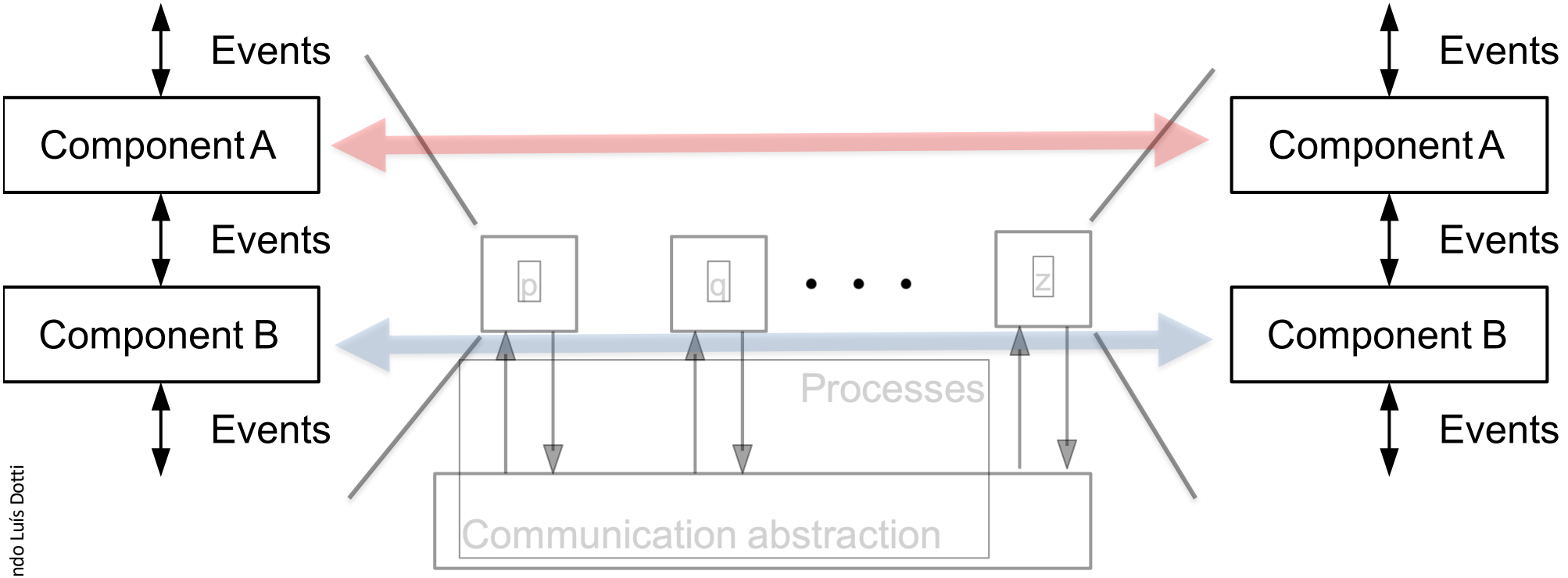
Processos e camadas



Módulos de mesmo nível interagem nos diversos processos para satisfazersuas propriedades.

Material baseado no livro: *Introduction to Reliable and Secure Distributed Programming* *Christian Cachin, Rachid Gerraoui, Luís Rodrigues*

Processos e camadas



Material baseado no livro: *Introduction to Reliable and Secure Distributed Programming* *Christian Cachin, Rachid Gerraoui, Luís Rodrigues*

Exemplo
Comunicação ponto a ponto

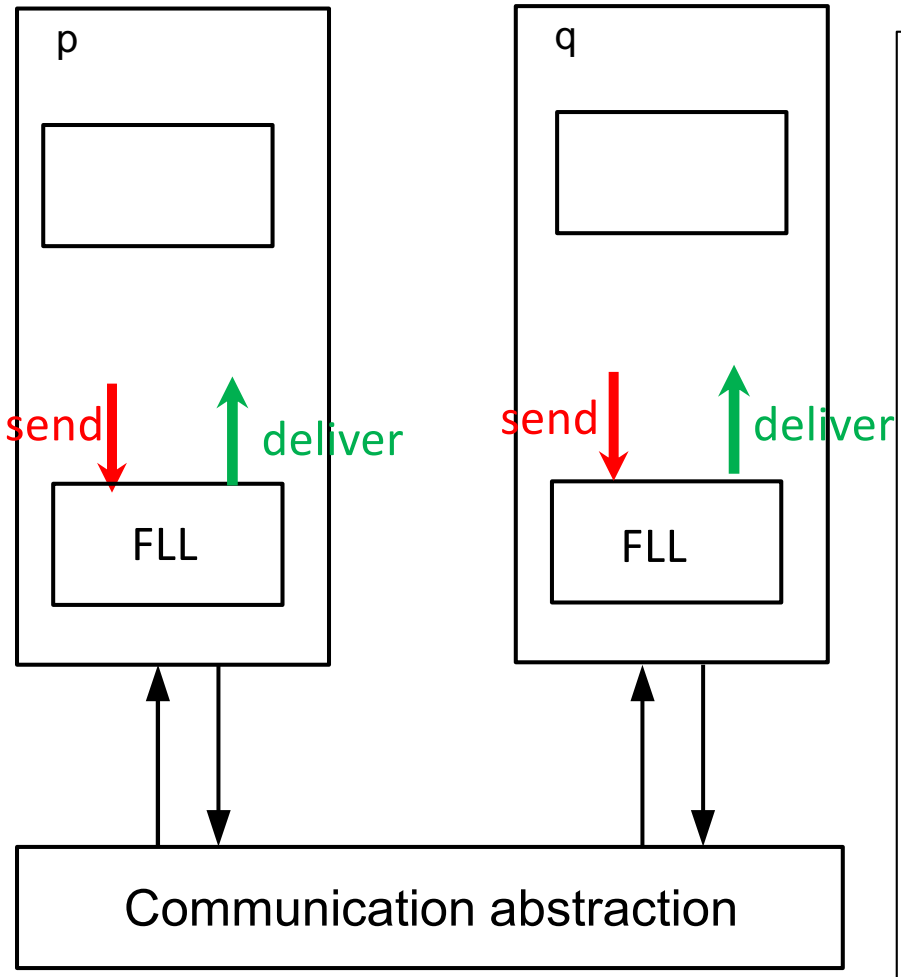
Material baseado no livro: *Introduction to Reliable and Secure Distributed Programming Christian Cachin, Rachid Gerraoui, Luís Rodrigues*

Comunicação ponto a ponto

- Tome-se a Internet como exemplo
  - os dados são fracionados em "pacotes", que são enviados ao destino, e no destino são remontados, se necessário
  - pacotes atravessam uma sucessão de "enlaces" e "roteadores" para chegarem ao destino, por exemplo a rede sem fio, a rede ethernet, a conexão de fibra ótica, etc.
  - enlaces podem ter problemas permantentes ou transientes
  - roteadores podem estar congestionados
  - pacotes podem ser perdidos ou entregues fora de ordem
  - como criar a funcionalidade de comunicação "perfeita" ?

Comunicação ponto a ponto

- Especificicação de link com "perda justa"



Modulo:

FairLossPointToPointLinks, *instância* ***fll.***

Eventos:

Request: [fll, Send | q, m]: solicita envio de m para q

Indicação: [fll, Deliver | p, m]: entrega mensagem m enviada por p

Propriedades

FLL1: Perda Justa: se um processo p infinitamente sends uma mensagem m para um processo q, então q delivers a mensagem um número infinito de vezes;

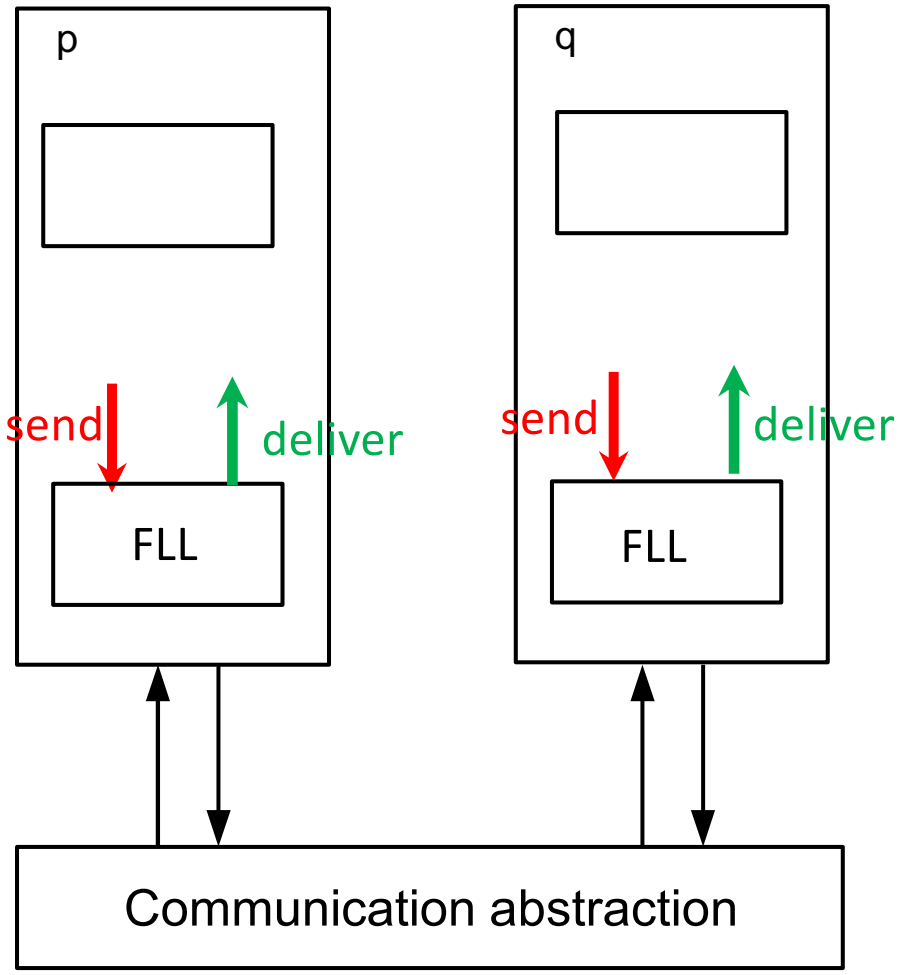
FLL2: Duplicação finita: se a mensagem é enviada um numero finito de vezes para p, entao p entrega um número finito de vezes

FLL3: Não criação: uma mensagem não é entregue se não tiver sido enviada

Material baseado no livro: *Introduction to Reliable and Secure Distributed Programming Christian Cachin, Rachid Gerraoui, Luís Rodrigues*

Comunicação ponto a ponto

- Especificicação de link com "perda justa"



Note que a especificação
RELACIONA EVENTOS GLOBAIS
nos módulos respectivos
DOS PROCESSOS DISTRIBUÍDOS

Cada módulo (instância) atua sobre dados
LOCAIS,
mas
a COMPOSIÇÃO dos módulos tem que
SATISFAZER as propriedades GLOBAIS
da especificação.

Material baseado no livro: *Introduction to Reliable and Secure Distributed Programming* *Christian Cachin, Rachid Gerraoui, Luís Rodrigues*

Comunicação ponto a ponto

- Abstração de rede
  - suponha que sua rede tem a capacidade de mandar mensagens ponto a ponto, com as seguintes propriedades representadas pelo módulo *FairLossPointToPointLinks*

Modulo:

FairLossPointToPointLinks, *instância* ***fll.***

Eventos:

Request: [fll, Send | q, m]: solicita envio de m para q

Indicação: [fll, Deliver | p, m]: entrega mensagem m enviada por p

Propriedades

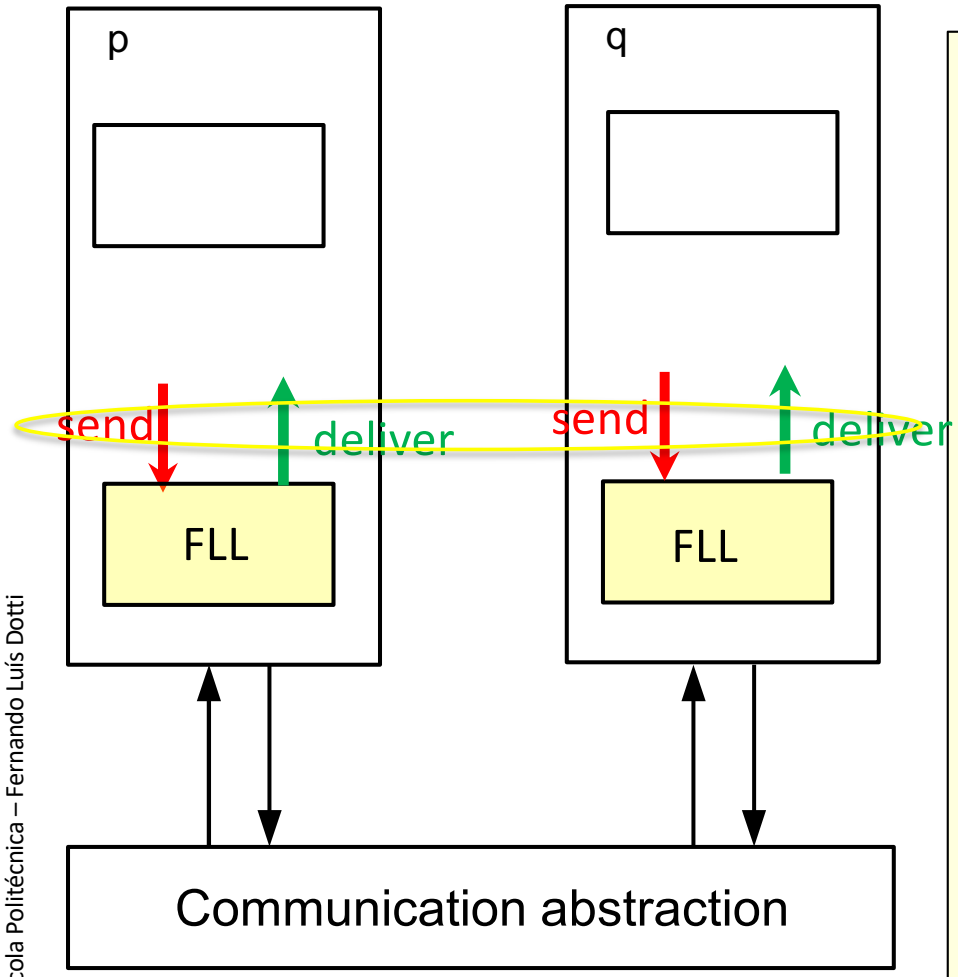
FLL1: Perda Justa: se um ***processo p*** *infinitamente sends uma mensagem m para* ***um processo q****, então* ***q*** *delivers a mensagem um número infinito de vezes;*

FLL2: Duplicação finita: se a mensagem é sent um numero finito de vezes para p, entao p deliver um número finito de vezes

FLL3: Não criação: uma mensagem não é deliver se não tiver sido sent

Material baseado no livro: *Introduction to Reliable and Secure Distributed Programming Christian Cachin, Rachid Gerraoui, Luís Rodrigues*

- **As propriedades especificam como os eventos distribuídos de send e deliver se relacionam!!!**



Modulo:

FairLossPointToPointLinks, *instância* ***fll.***

Eventos:

Request: [fll, Send | q, m]: solicita envio de m para q

Indicação: [fll, Deliver | p, m]: entrega mensagem m enviada por p

Propriedades

FLL1: Perda Justa: se um processo p infinitamente sends uma mensagem m para um processo q, então q delivers a mensagem um número infinito de vezes;

FLL2: Duplicação finita: se a mensagem é enviada um numero finito de vezes para p, entao p entrega um número finito de vezes

FLL3: Não criação: uma mensagem não é entregue se não tiver sido enviada

Material baseado no livro: *Introduction to Reliable and Secure Distributed Programming Christian Cachin, Rachid Gerraoui, Luís Rodrigues*

Comunicação ponto a ponto

- Suponha que, voce dispõe de um módulo **FLL** e deve **criar a abstração de um link perfeito** (perfect link - PL)

Modulo:

PerfectPointToPointLinks, *instancia* ***pl***

Eventos:

Request: [pl, Send | q, m]: solicita envio de m para q

Indicação: [pl, Deliver | p, m]: entrega mensagem m enviada por p

Propriedades

PL1: Entrega Confiável: se um processo sends uma mensagem m para um processo q, então q entregará (delivers) a mensagem em algum momento;

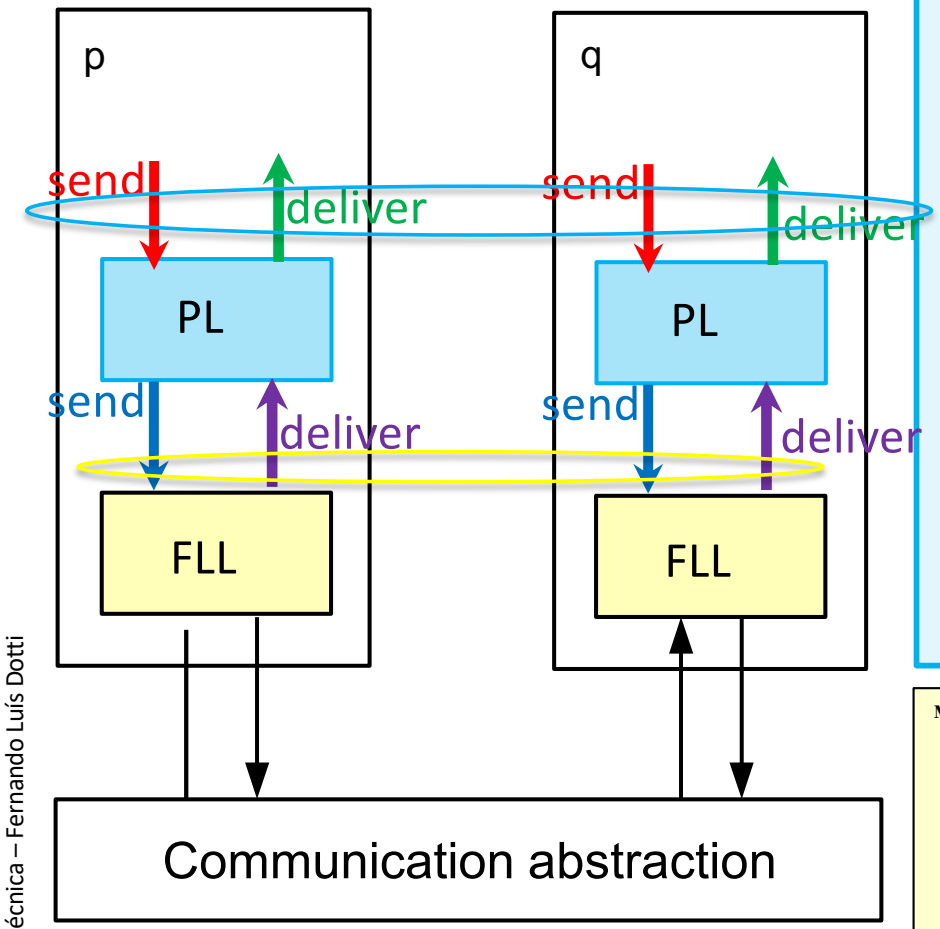
PL2: Não Duplicação: uma mensagem não é entregue por um processo mais de uma vez

PL3: Não criação: uma mensagem não é entregue se não tiver sido enviada

Material baseado no livro: *Introduction to Reliable and Secure Distributed Programming Christian Cachin, Rachid Gerraoui, Luís Rodrigues*

Comunicação ponto a ponto

- PL sobre FLL



Modulo:

PerfectPointToPointLinks, *instancia pl*

Eventos de interface (usados pelo módulo superior)**:**

Request: [pl, Send | q, m]: solicita envio de m para q

Indicação: [pl, Deliver | p, m]: entrega mensagem m enviada por p

Propriedades

PL1: Entrega Confiável: se um processo sends uma mensagem m para um processo q, então q entregara (delivers) a mensagem em algum momento;

PL2: Não Duplicação: uma mensagem não é entregue por um processo mais de uma vez

PL3: Não criação: uma mensagem não é entregue se não tiver sido enviada

Modulo:

FairLossPointToPointLinks

Eventos de interface (usados pelo módulo superior):

Request: [fll, Send | q, m]: solicita envio de m para q

Indicação: [fll, Deliver | p, m]: entrega mensagem m enviada por p

Propriedades

FLL1: Perda Justa: se um processo p infinitamente sends uma mensagem m para um processo q, então q delivers a mensagem um número infinito de vezes;

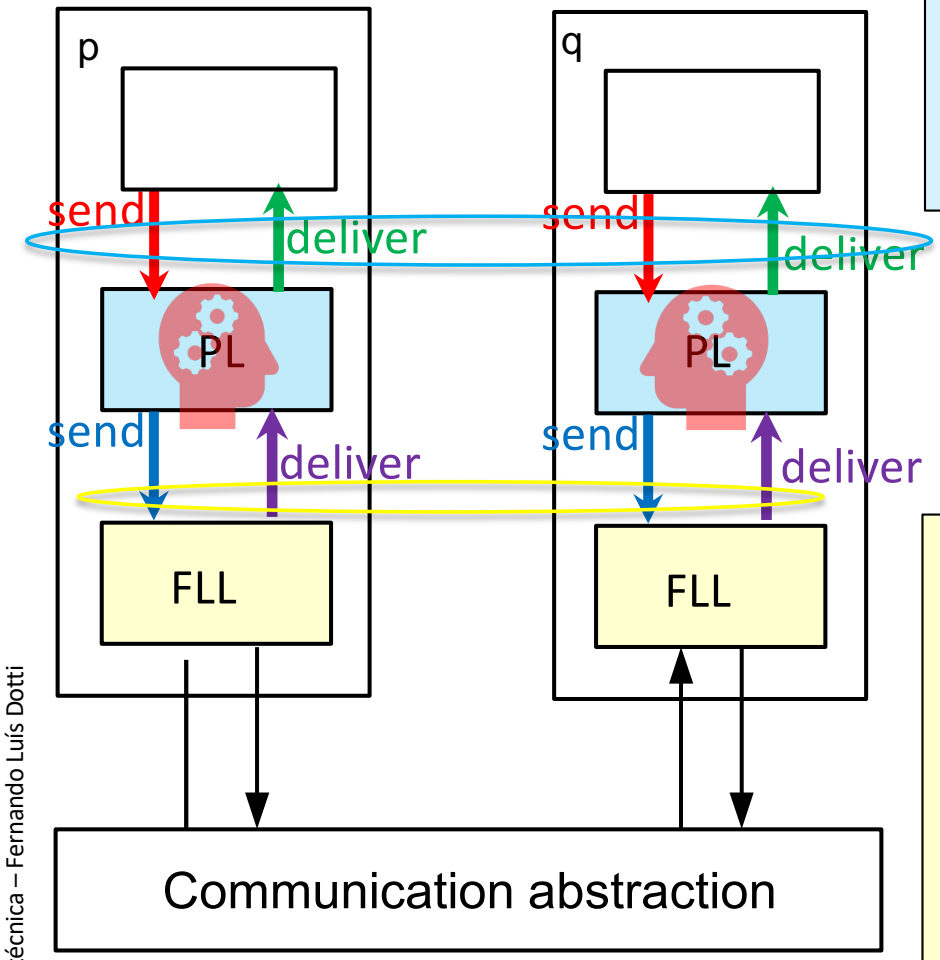
FLL2: Duplicação finita: se a mensagem é enviada um numero finito de vezes para p, entao p entrega um número finito de vezes

FLL3: Não criação: uma mensagem não é entregue se não tiver sido enviada

Material baseado no livro: *Introduction to Reliable and Secure Distributed Programming* *Christian Cachin, Rachid Gerraoui, Luís Rodrigues*

Comunicação ponto a ponto

- PL sobre FLL



Modulo:

PerfectPointToPointLinks, *instancia pl*

Eventos de interface (usados pelo módulo superior)**:**

Request: [pl, Send | q, m]: solicita envio de m para q

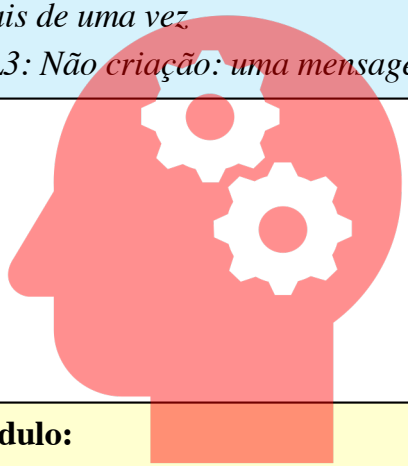
Indicação: [pl, Deliver | p, m]: entrega mensagem m enviada por p

Propriedades

PL1: Entrega Confiável: se um processo sends uma mensagem m para um processo q, então q entregara (delivers) a mensagem em algum momento;

PL2: Não Duplicação: uma mensagem não é entregue por um processo mais de uma vez

PL3: Não criação: uma mensagem não é entregue se não tiver sido enviada



Modulo:

FairLossPointToPointLinks

Eventos de interface (usados pelo módulo superior):

Request: [fll, Send | q, m]: solicita envio de m para q

Indicação: [fll, Deliver | p, m]: entrega mensagem m enviada por p

Propriedades

FLL1: Perda Justa: se um processo p infinitamente sends uma mensagem m para um processo q, então q delivers a mensagem um número infinito de vezes;

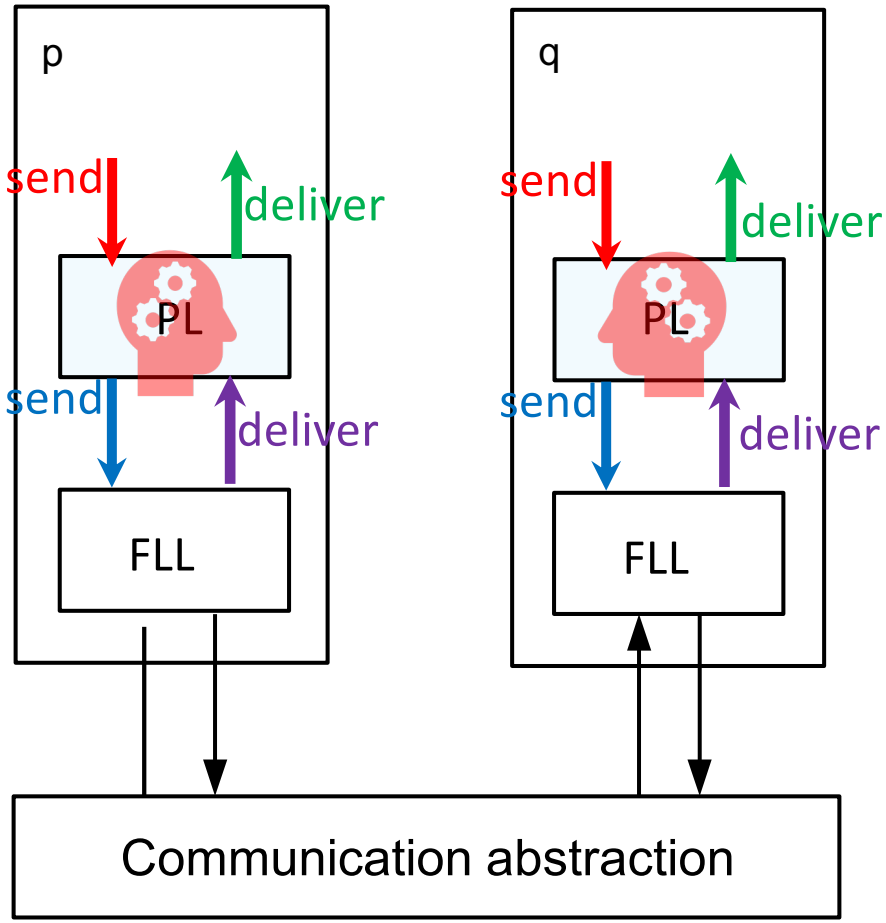
FLL2: Duplicação finita: se a mensagem é enviada um numero finito de vezes para p, entao p entrega um número finito de vezes

FLL3: Não criação: uma mensagem não é entregue se não tiver sido enviada

Material baseado no livro: *Introduction to Reliable and Secure Distributed Programming* *Christian Cachin, Rachid Gerraoui, Luís Rodrigues*

Comunicação ponto a ponto

- PL sobre FLL



Algoritmo para PL: repete indefinidamente, elimina duplicatas

upon event [pl, Init] do

upon event [pl, Send | q, m] do

trigger [fll, Send | q, m]

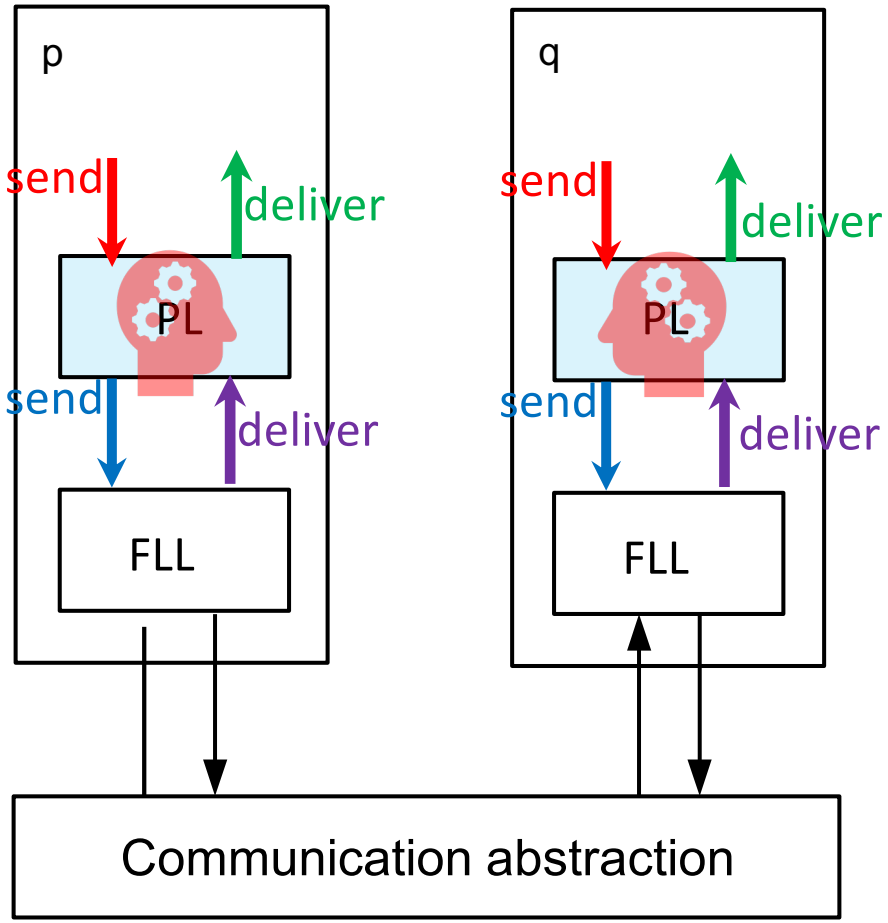
upon event [fll, Deliver | p, m] do

trigger [pl, Deliver | p, m]

Material baseado no livro: *Introduction to Reliable and Secure Distributed Programming* *Christian Cachin, Rachid Gerraoui, Luís Rodrigues*

Comunicação ponto a ponto

- PL sobre FLL



Algoritmo para PL: repete indefinidamente, elimina duplicatas

enviadas: conjunto de mensagens enviadas

entregues: conjunto de mensagens entregues

DeltaT

upon event [pl, Init] do

entregues :={}

enviadas := {}

DeltaT := tempo para repetir mensagens

upon event [pl, Send | q, m] do

trigger [fll, Send | q, m]

enviadas := enviadas U {(q,m)}

upon event [fll, Deliver | p, m] do

se m não pertence a entregues

então entregues := entregues U { m }

trigger [pl, Deliver | p, m]

upon event [Timeout] do

para todo (q, m) pertencente a enviadas

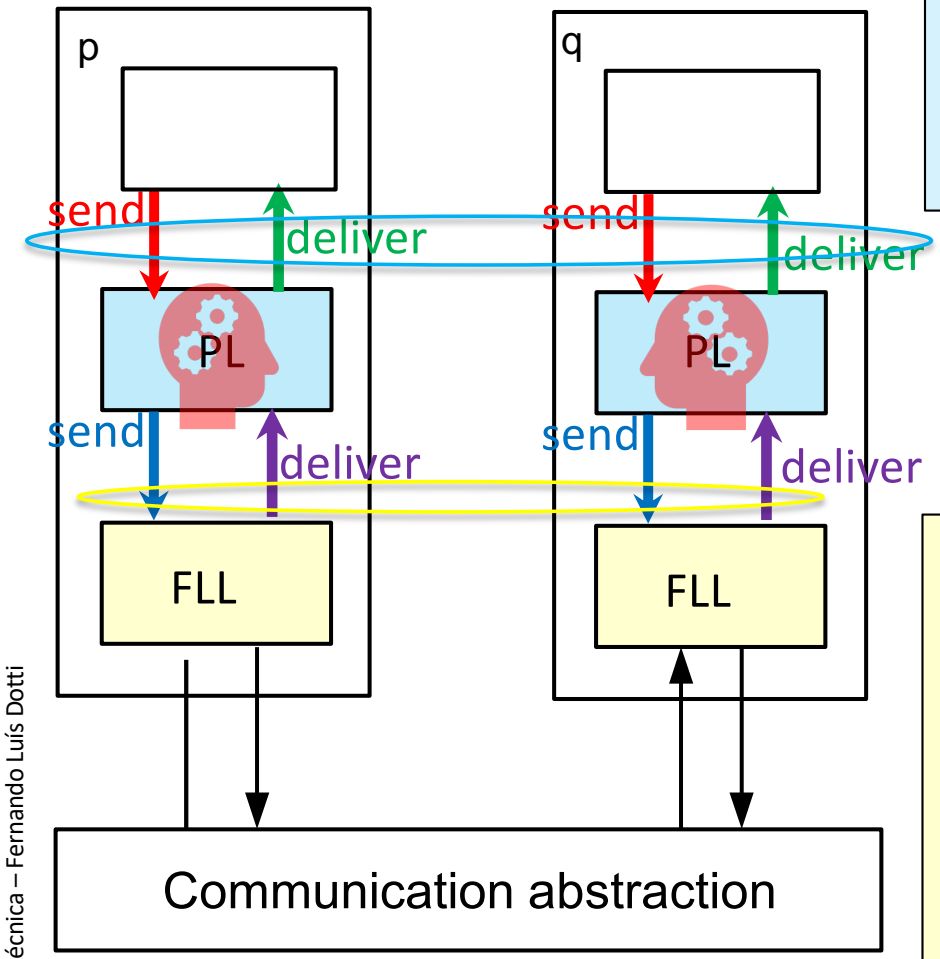
trigger [fll, Send | q, m]

reinicia timer para gerar Timeout em DeltaT

Material baseado no livro: *Introduction to Reliable and Secure Distributed Programming* *Christian Cachin, Rachid Gerraoui, Luís Rodrigues*

Comunicação ponto a ponto

- PL sobre FLL



Modulo:

PerfectPointToPointLinks, *instancia pl*

Eventos de interface (usados pelo módulo superior)**:**

Request: [pl, Send | q, m]: solicita envio de m para q

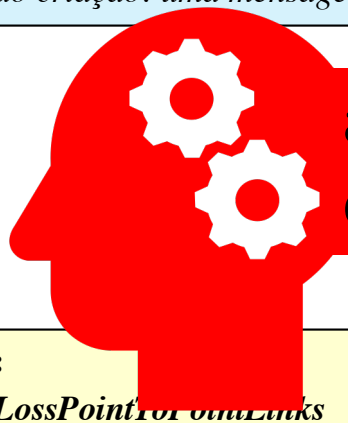
Indicação: [pl, Deliver | p, m]: entrega mensagem m enviada por p

Propriedades

PL1: Entrega Confiável: se um processo sends uma mensagem m para um processo q, então q entregara (delivers) a mensagem em algum momento;

PL2: Não Duplicação: uma mensagem não é entregue por um processo mais de uma vez

PL3: Não criação: uma mensagem não é entregue se não tiver sido enviada



argumentação com o algoritmo proposto

Modulo:

FairLossPointToPointLinks

Eventos de interface (usados pelo módulo superior):

Request: [fll, Send | q, m]: solicita envio de m para q

Indicação: [fll, Deliver | p, m]: entrega mensagem m enviada por p

Propriedades

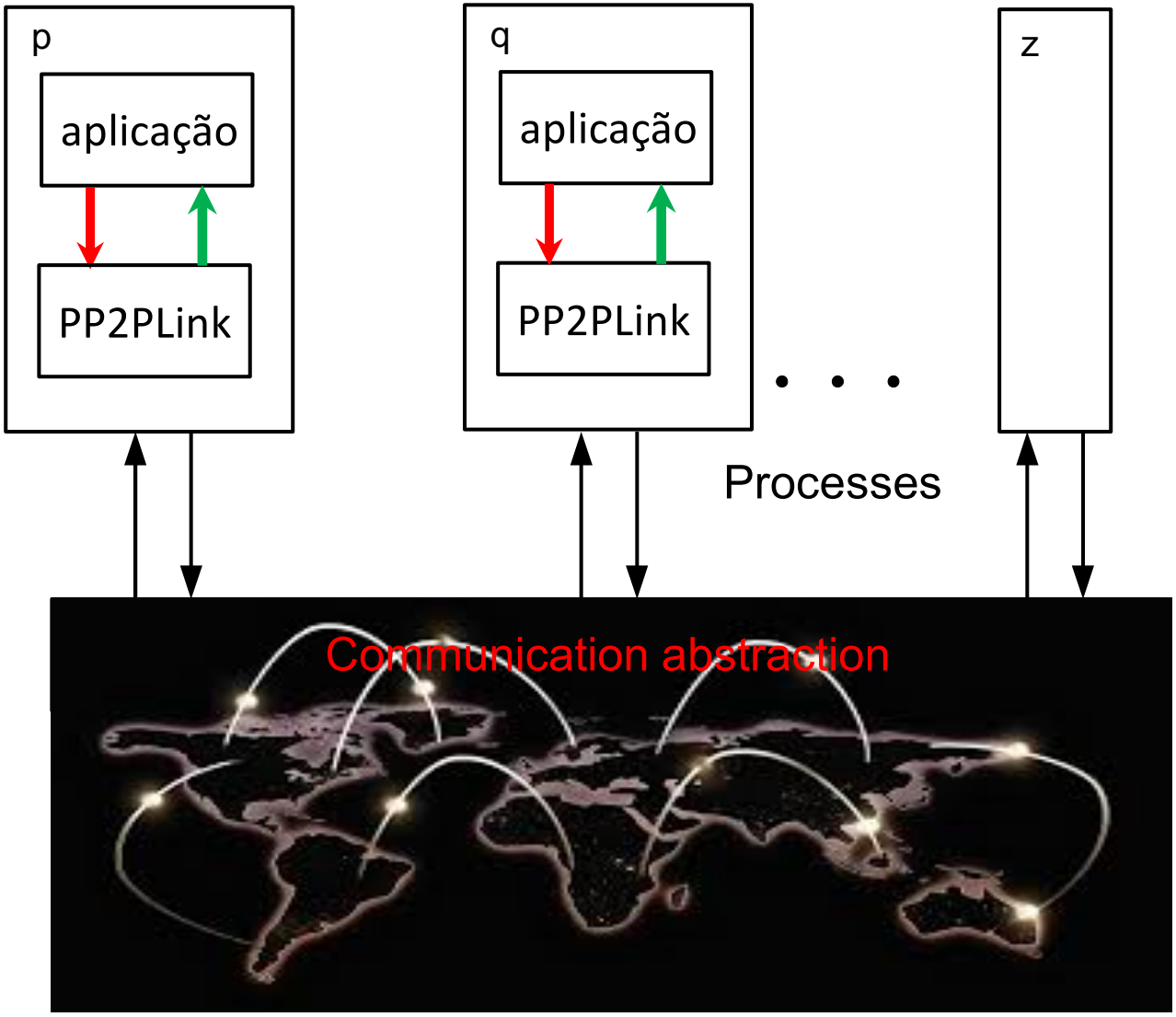
FLL1: Perda Justa: se um processo p infinitamente sends uma mensagem m para um processo q, então q delivers a mensagem um número infinito de vezes;

FLL2: Duplicação finita: se a mensagem é enviada um numero finito de vezes para p, entao p entrega um número finito de vezes

FLL3: Não criação: uma mensagem não é entregue se não tiver sido enviada

Material baseado no livro: *Introduction to Reliable and Secure Distributed Programming* *Christian Cachin, Rachid Gerraoui, Luís Rodrigues*

Comunicação ponto a ponto



abstração:
conjunto de processos trocam mensagens 1:1 de forma transparente sobre rede de computadores

Material baseado no livro: *Introduction to Reliable and Secure Distributed Programming* *Christian Cachin, Rachid Gerraoui, Luís Rodrigues*