Reator

- Problema C10K
- Questões do Sistema Operacional
- Programação Orientada a Eventos
- Detalhes do padrão

C10K

Milhares de clientes concorrentes (ex: 20K)
Será que os sistemas modernos escalam?

Problemas com:

- Hardware
- Sistema Operacional
- Aplicação

C10K

Forças atuantes:

- **Disponibilidade**
- **Eficiência**
- **Simplicidade do Programa**
- **Capacidade de adaptação**
- **Portabilidade**

Hardware

Servidor:
2GHz 2GB RAM 1Gbits/s = R$2000

20 000 clientes

Cada cliente:
100KHz, 100KB e 50Kbits/s = R$ 0,10

Sistema Operacional: Estratégias de E/S

- Quando e como tratar o problema de chamadas múltiplas de E/S de uma linha de execução única?
  - “Desencana”! Use múltiplas linhas de execução.
  - Use Chamadas que *não bloqueiam* e notificam estado *de prontidão*
  - Use Chamadas assíncronas

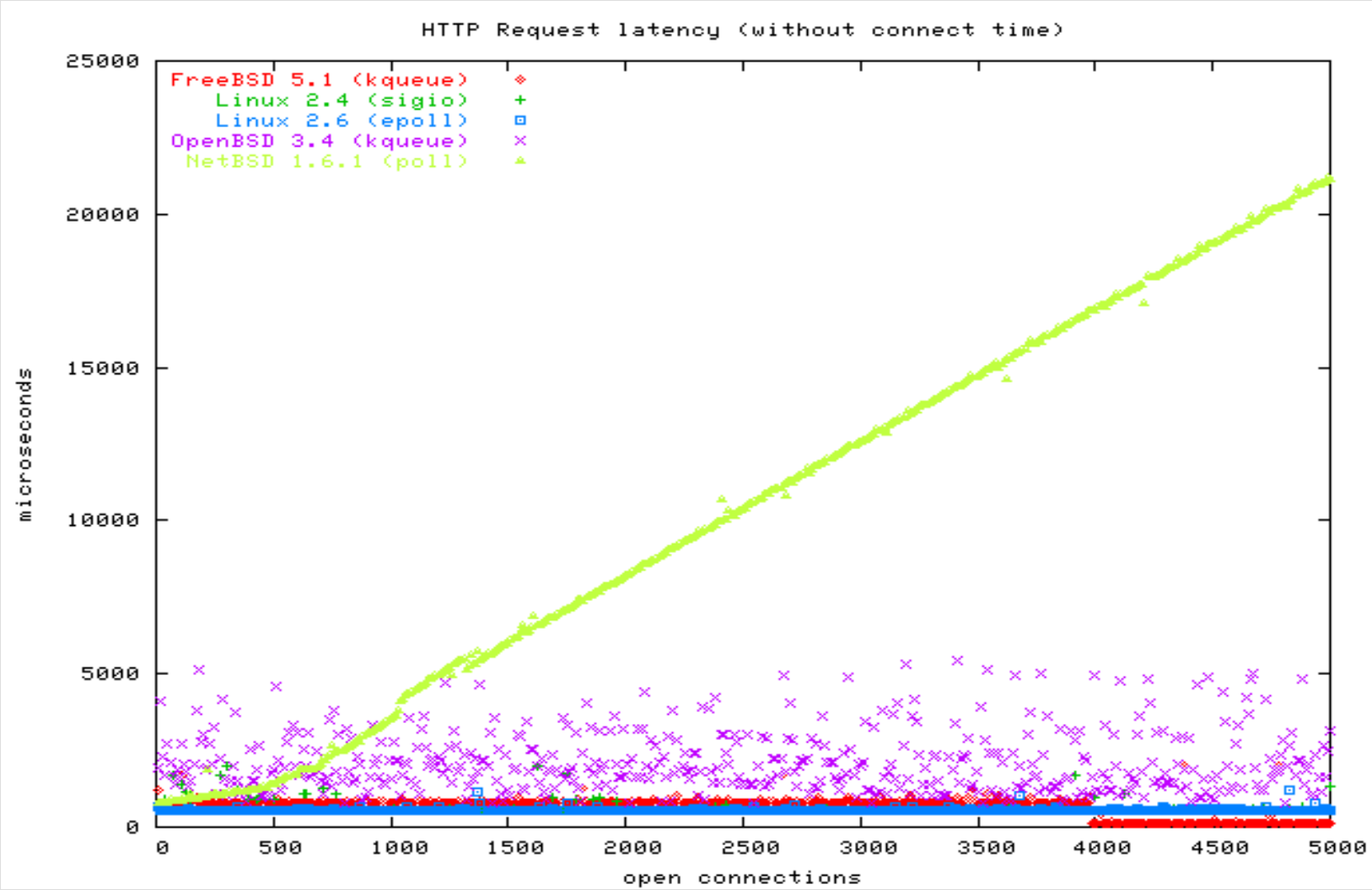
Sistema Operacional: Estratégias de E/S

- Servir muitos clientes com uma linha de execução
  - usar E/S sem bloqueador
    - Notificação de prontidão no nível-disparador
    - Notificação de mudança no estado de prontidão
  - usar E/S assíncrona
- Servir um cliente com uma linha de execução
- Colocar o código no núcleo do SO

Por que usar linha de execução única (single-thread)

- Eficiência
- Simplicidade na programação
- Portabilidade

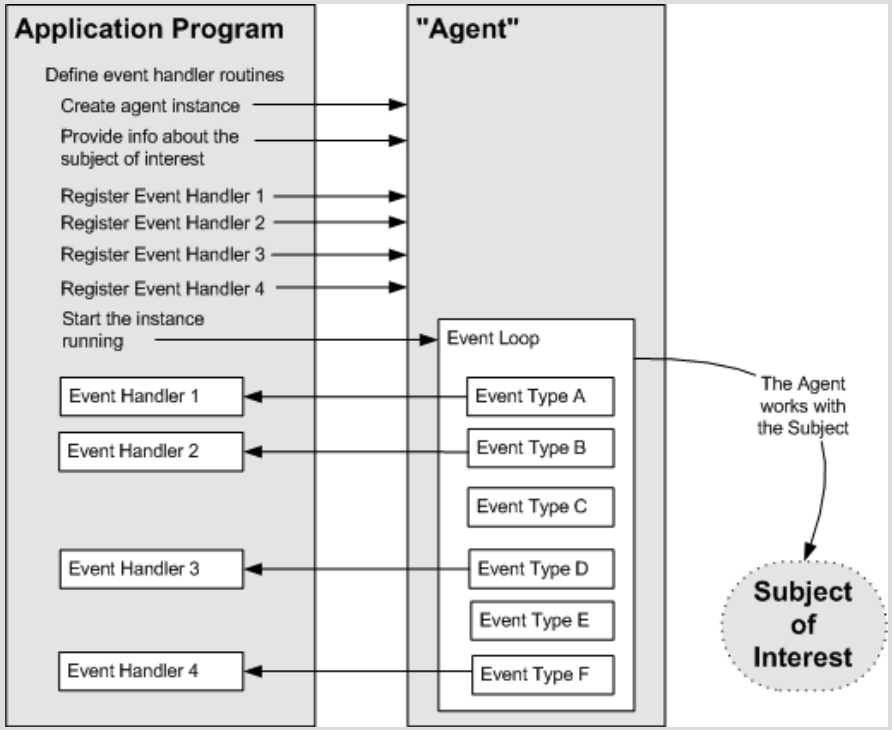
Por que usar notificação da mudança do estado de prontidão



Aplicação: Padrão Reator

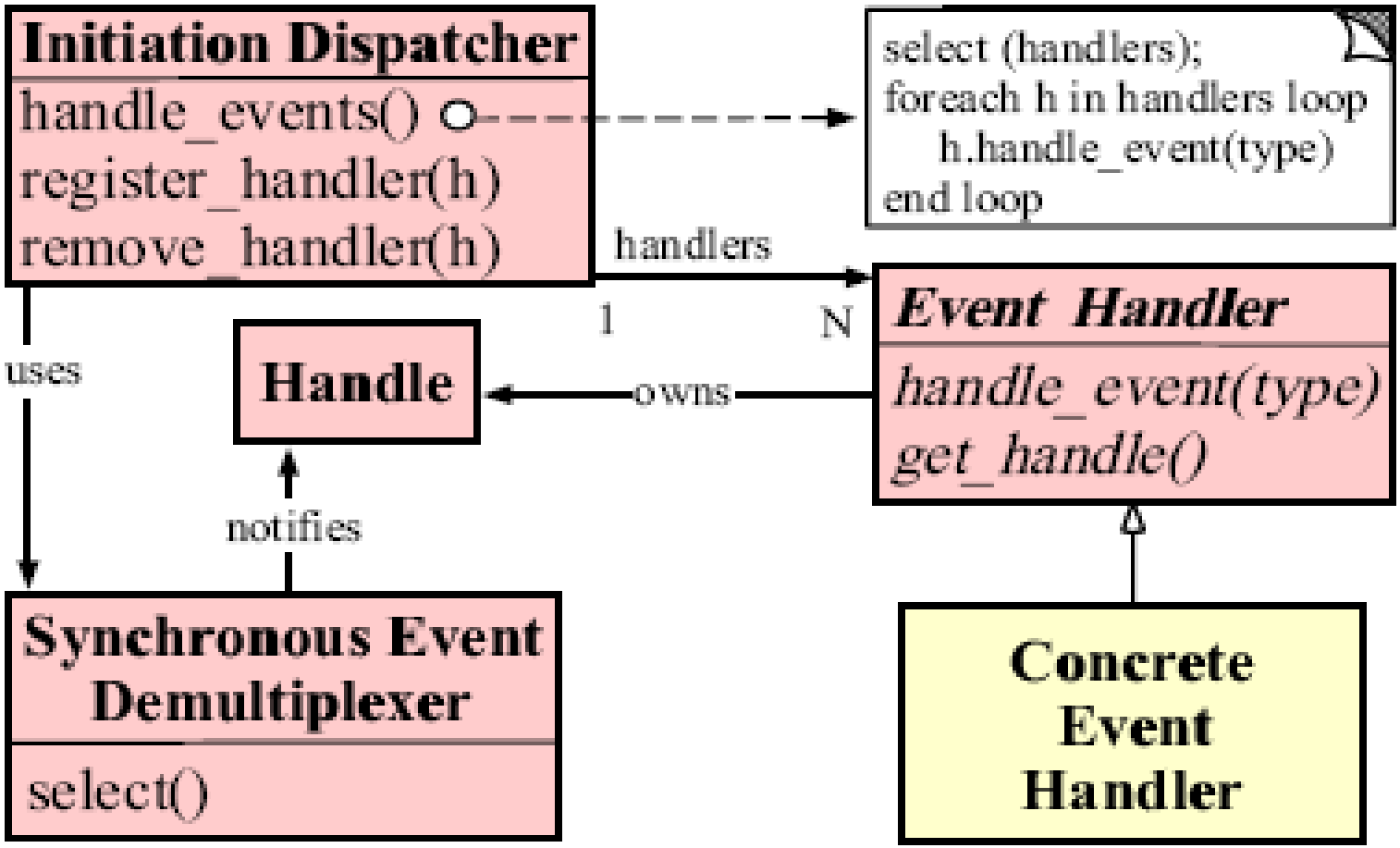
- O conceito assemelha-se com uma composição do padrão observador adaptado ao mecanismo de demultiplexação.
- Um objeto despachante assume duas funções: integra o demultiplexador síncrono de eventos e despacha os eventos para os manipuladores correspondentes (paradigma de programação orientado a eventos).

Programação Orientada a Eventos

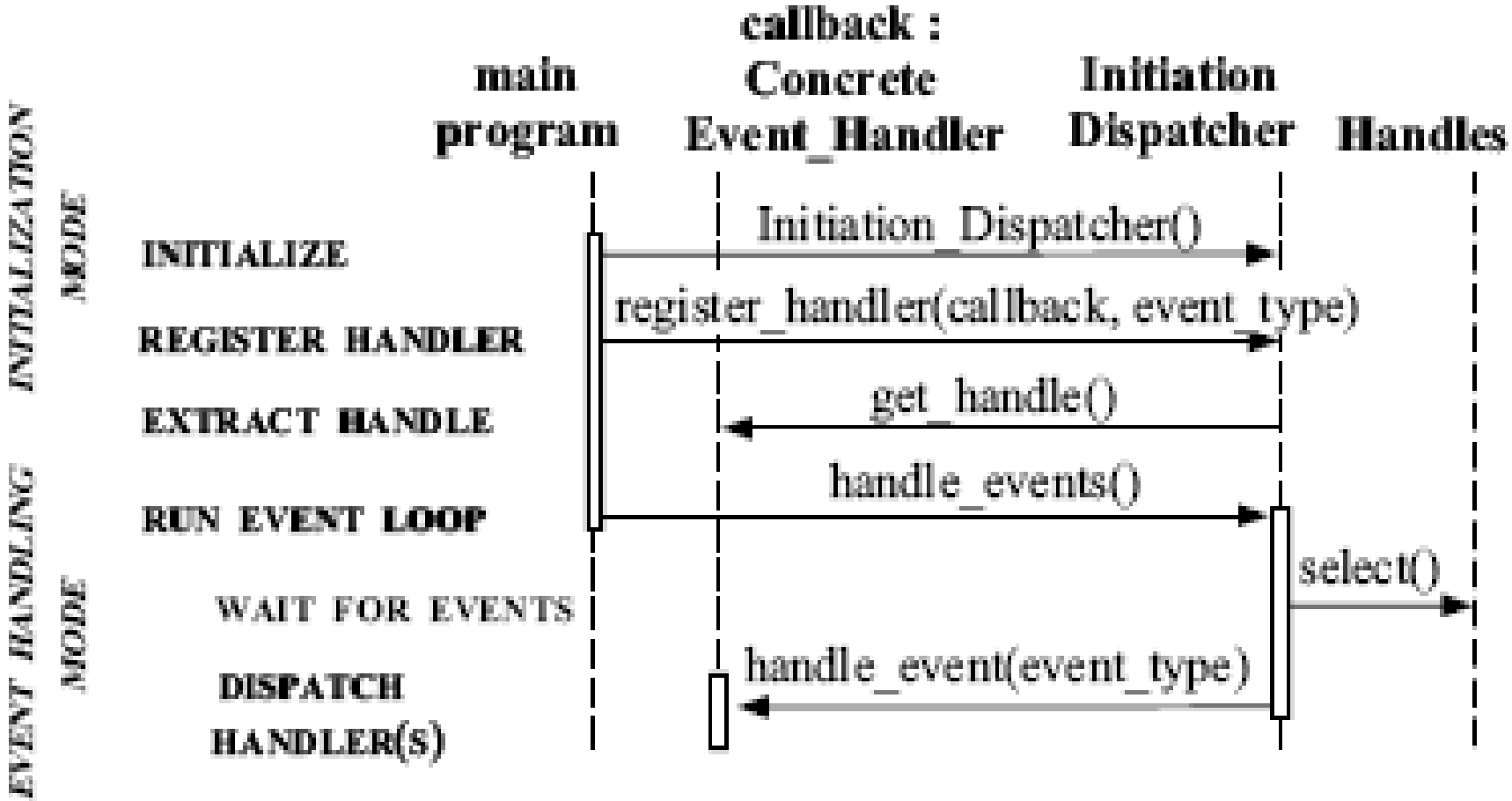


- Estrutura
  - agente ou despachante
  - manipuladores de evento (event-handlers)
  - evento

Reator - Estrutura



Reator - Dinâmica



Benefícios

- **Separação de Conceitos:** O padrão Reator desacopla mecanismo de demultiplexação e despache de métodos de funcionalidades específicas da aplicação;
- **Melhora modularidade, reuso e facilidade de configuração da aplicação orientada a eventos:** O padrão desacopla funcionalidades da aplicação em classes separadas.;
- **Melhora a portabilidade da aplicação**
- **Melhora o controle da concorrência de alta granularidade:** O padrão Reator serializa a chamada dos manipuladores de evento.

Malefícios

- Possibilidade de aplicação restrita: o SO precisa tratar Referências (Handles)
- Não Preemptivo
- Difícil de depurar

Padrões Relacionados

- Observador (*Observer*)
- Cadeia de Responsabilidade
- Proator (Proactor)
- Façade

Referências

- Reactor: Douglas C. Schmidt
  "Pattern Languages of Program Design" ISBN 0-201-6073-4, editado por Jim Coplien e Douglas C. Schmidt e publicado por Addison-Wesley, 1995.
- The C10K problem: Dan Kegel
  http://www.kegel.com/c10k.html#aio
- Benchmarking BSD and Linux: Felix von Leitner
  http://bulk.fefe.de/scalability/
- Introduction to Event-Driven Programming: Stephen Ferg
  http://www.ferg.org/projects/ferg-event_driven_programming.html