Organização e Recuperação de Informação: Crawling

Marcelo K. Albertini

Faculdade de Computação, Universidade Federal de Uberlândia

Conteúdo

1 Um crawler simples

2 Um crawler real

É difícil projetar um crawler?

- Buscadores devem capturar e acumular documentos da web

É difícil projetar um crawler?

- Buscadores devem capturar e acumular documentos da web
- Pegar conteúdo de docs é mais fácil para outros sistemas de ORI

É difícil projetar um crawler?

- Buscadores devem capturar e acumular documentos da web
- Pegar conteúdo de docs é mais fácil para outros sistemas de ORI
  - E.g., indexar arquivos no seu disco rígido: somente fazer uma recursão nos diretórios no computador

É difícil projetar um crawler?

- Buscadores devem capturar e acumular documentos da web
- Pegar conteúdo de docs é mais fácil para outros sistemas de ORI
  - E.g., indexar arquivos no seu disco rígido: somente fazer uma recursão nos diretórios no computador
- Para ORI na web, pegar conteúdo de docs leva mais tempo

...

É difícil projetar um crawler?

- Buscadores devem capturar e acumular documentos da web
- Pegar conteúdo de docs é mais fácil para outros sistemas de ORI
  - E.g., indexar arquivos no seu disco rígido: somente fazer uma recursão nos diretórios no computador
- Para ORI na web, pegar conteúdo de docs leva mais tempo . . .
- . . . devido à latência

É difícil projetar um crawler?

- Buscadores devem capturar e acumular documentos da web
- Pegar conteúdo de docs é mais fácil para outros sistemas de ORI
  - E.g., indexar arquivos no seu disco rígido: somente fazer uma recursão nos diretórios no computador
- Para ORI na web, pegar conteúdo de docs leva mais tempo . . .
- . . . devido à latência
- Crawler (ou robô de indexação ou rastreador web) é o programa que captura documentos para um buscador indexar

É difícil projetar um crawler?

- Buscadores devem capturar e acumular documentos da web
- Pegar conteúdo de docs é mais fácil para outros sistemas de ORI
  - E.g., indexar arquivos no seu disco rígido: somente fazer uma recursão nos diretórios no computador
- Para ORI na web, pegar conteúdo de docs leva mais tempo . . .
- . . . devido à latência
- Crawler (ou robô de indexação ou rastreador web) é o programa que captura documentos para um buscador indexar
- É difícil fazer um?

Operação de um crawler simples

`URL = Uniform Resource Locator` – "endereço da página"

- Inicializar fila com URLs de páginas iniciais

Operação de um crawler simples

`URL = Uniform Resource Locator` – "endereço da página"

- Inicializar fila com URLs de páginas iniciais
- Repetir

Operação de um crawler simples

`URL = Uniform Resource Locator` – "endereço da página"

- Inicializar fila com URLs de páginas iniciais
- Repetir
  - Pegar URL da fila

Operação de um crawler simples

`URL = Uniform Resource Locator` – "endereço da página"

- Inicializar fila com URLs de páginas iniciais
- Repetir
  - Pegar URL da fila
  - Capturar e processar página

Operação de um crawler simples

`URL = Uniform Resource Locator` – "endereço da página"

- Inicializar fila com URLs de páginas iniciais
- Repetir
  - Pegar URL da fila
  - Capturar e processar página
  - Extrair URLs da página

Operação de um crawler simples

`URL = Uniform Resource Locator` – "endereço da página"

- Inicializar fila com URLs de páginas iniciais
- Repetir
  - Pegar URL da fila
  - Capturar e processar página
  - Extrair URLs da página
  - Adicionar URLs para fila

Operação de um crawler simples

`URL = Uniform Resource Locator` – "endereço da página"

- Inicializar fila com URLs de páginas iniciais
- Repetir
  - Pegar URL da fila
  - Capturar e processar página
  - Extrair URLs da página
  - Adicionar URLs para fila
- Premissa fundamental: a web é bem conectada

Exercício: qual é o problema deste crawler?

```
filaUrls := (urls de inicialização bem escolhidas)
enquanto filaUrls não está vazia:
  url := filaUrls.pegarUltimoEremover()
  pagina := url.capturar()
  urlsCapturados.adicionar(url)
  novosUrls := pagina.extrairUrls()
  para cada url em novosUrls:
    se url não está em urlsCapturados e nem em filaUrls:
      filaUrls.adicionar(url)
  fim para
  adicionarParaIndiceInvertido(pagina)
fim enquanto
```

Quais são os problemas desse crawler simples

- Para grande quantidade de docs é necessário distribuir.
- Não podemos indexar tudo: necessitamos amostrar.
- Duplicatas: precisamos usar detecção de duplicatas
- Spam e armadilhas de aranhas: usar detecção de spam
- Cortesia: necessário esparsar requisições para evitar sobrecarregar um site
- Recência: necessário capturar periodicamente
  - Devido ao tamanho da web, podemos fazer re-capturas apenas para um pequeno subconjunto de páginas
  - Outra vez, problema de amostragem ou prioritização

Magnitude do problema de crawling

- Para capturar 20,000,000,000 páginas em um mês . . .

Magnitude do problema de crawling

- Para capturar 20,000,000,000 páginas em um mês . . .
- . . . precisamos obter quase 8000 páginas por segundo

Magnitude do problema de crawling

- Para capturar 20,000,000,000 páginas em um mês . . .
- . . . precisamos obter quase 8000 páginas por segundo
- Na verdade: muito mais pois muitas da páginas serão duplicatas, indisponíveis, spam etc.

O que um crawler deve fazer?

Cortesia

O que um crawler deve fazer?

Cortesia

- Não sobrecarregar um site

O que um crawler deve fazer?

Cortesia

- Não sobrecarregar um site
- Pegar páginas apenas autorizadas: robots.txt

O que um crawler deve fazer?

Cortesia

- Não sobrecarregar um site
- Pegar páginas apenas autorizadas: robots.txt
- `Protocolo de exclusão de robôs`

Robustez

O que um crawler deve fazer?

Cortesia

- Não sobrecarregar um site
- Pegar páginas apenas autorizadas: robots.txt
- `Protocolo de exclusão de robôs`
- Sitemaps: `Protocolo de inclusão de robôs`

Robustez

- Resistência a "armadilhas", duplicatas, páginas e sites muito grandes, páginas dinâmicas etc

Robots.txt

- Protocolo para “robots” acessar um website

Robots.txt

- Protocolo para “robots” acessar um website
- Exemplos:

Robots.txt

- Protocolo para "robots" acessar um website
- Exemplos:
  - User-agent: *
    Disallow: /yoursite/temp/

Robots.txt

- Protocolo para "robots" acessar um website
- Exemplos:
  - User-agent: *
    Disallow: /yoursite/temp/
  - User-agent: searchengine
    Disallow: /

Robots.txt

- Protocolo para "robots" acessar um website
- Exemplos:
  - User-agent: *
    Disallow: /yoursite/temp/
  - User-agent: searchengine
    Disallow: /
- Importante: armazenar o arquivo robots.txt de cada site que estamos acessando

Exemplo de robots.txt (wikipedia)

```
# advertising-related bots:
User-agent: Mediapartners-Google*
Disallow: /
# Wikipedia work bots:
User-agent: IsraBot
Disallow:
User-agent: Orthogaffe
Disallow:
# Crawlers that are kind enough to obey, but which we'd rat
# unless they're feeding search engines.
User-agent: UbiCrawler
Disallow: /
...
```

```
# Friendly, low-speed bots are welcome viewing article page
# dynamically-generated pages please.

User-agent: *
Allow: /w/api.php?action=mobileview&
Disallow: /w/
Disallow: /trap/
Disallow: /wiki/Especial:Search
Disallow: /wiki/Especial%3ASearch
```

O que todo bom crawler de web deve fazer

- Capacidade de operação distribuída

O que todo bom crawler de web deve fazer

- Capacidade de operação distribuída
- Ser escalável: capacidade de aumentar taxa de captura por meio da adição de mais máquinas

O que todo bom crawler de web deve fazer

- Capacidade de operação distribuída
- Ser escalável: capacidade de aumentar taxa de captura por meio da adição de mais máquinas
- Capturar primeiro as páginas de melhor qualidade

O que todo bom crawler de web deve fazer

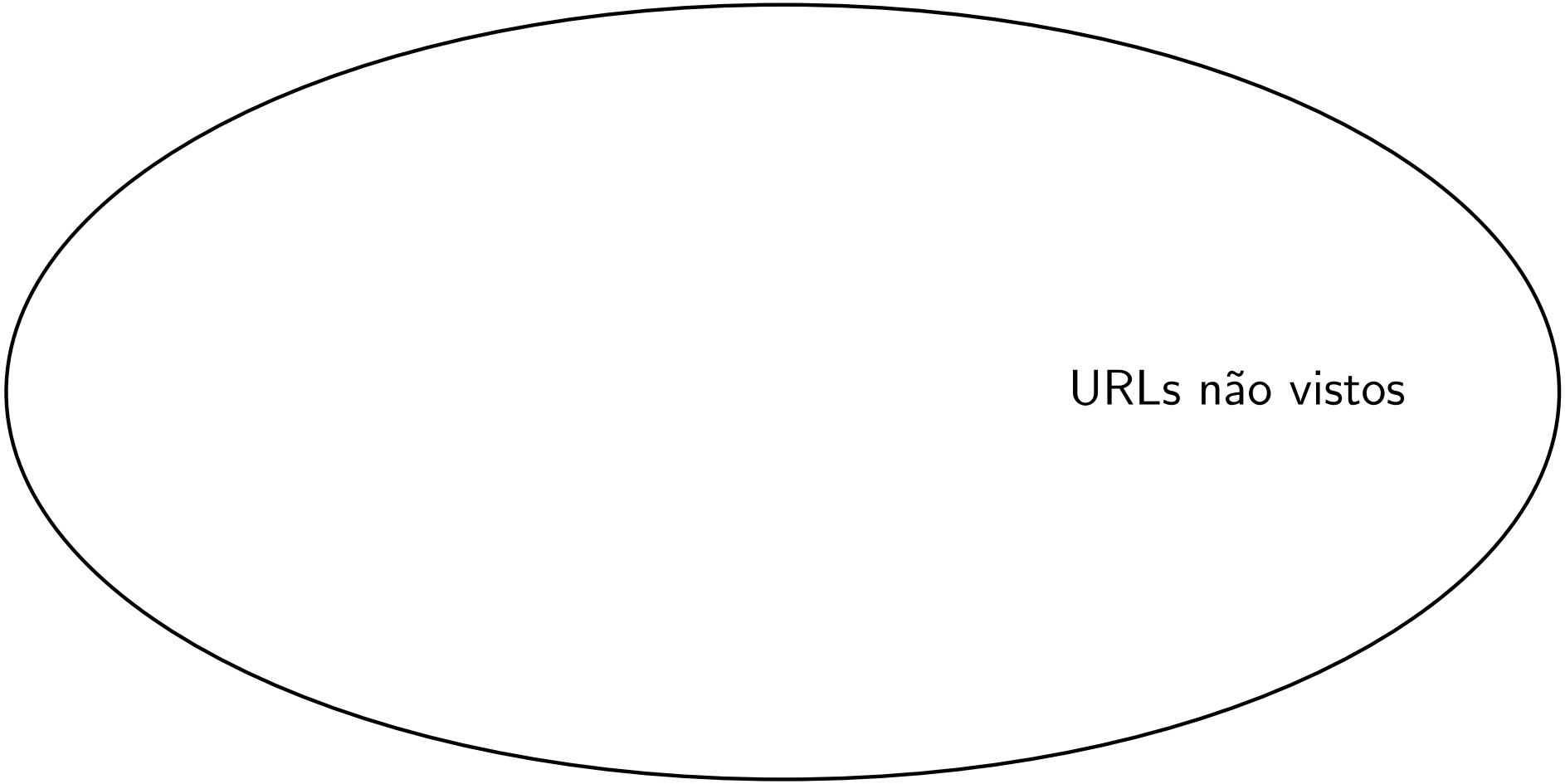
- Capacidade de operação distribuída
- Ser escalável: capacidade de aumentar taxa de captura por meio da adição de mais máquinas
- Capturar primeiro as páginas de melhor qualidade
- Operação contínua: obter versões recentes das páginas capturadas anteriormente

Conteúdo

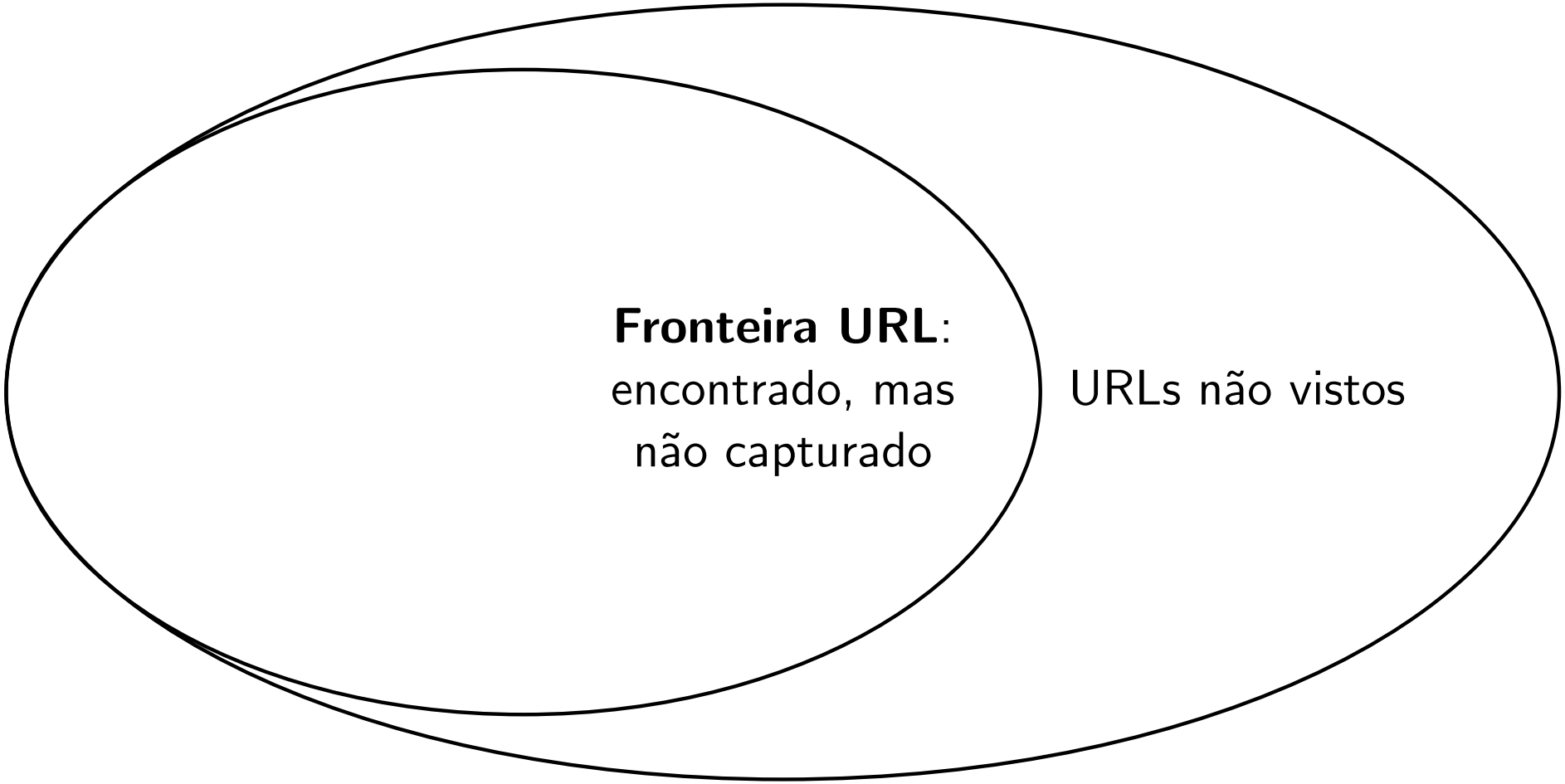
1 Um crawler simples

2 Um crawler real

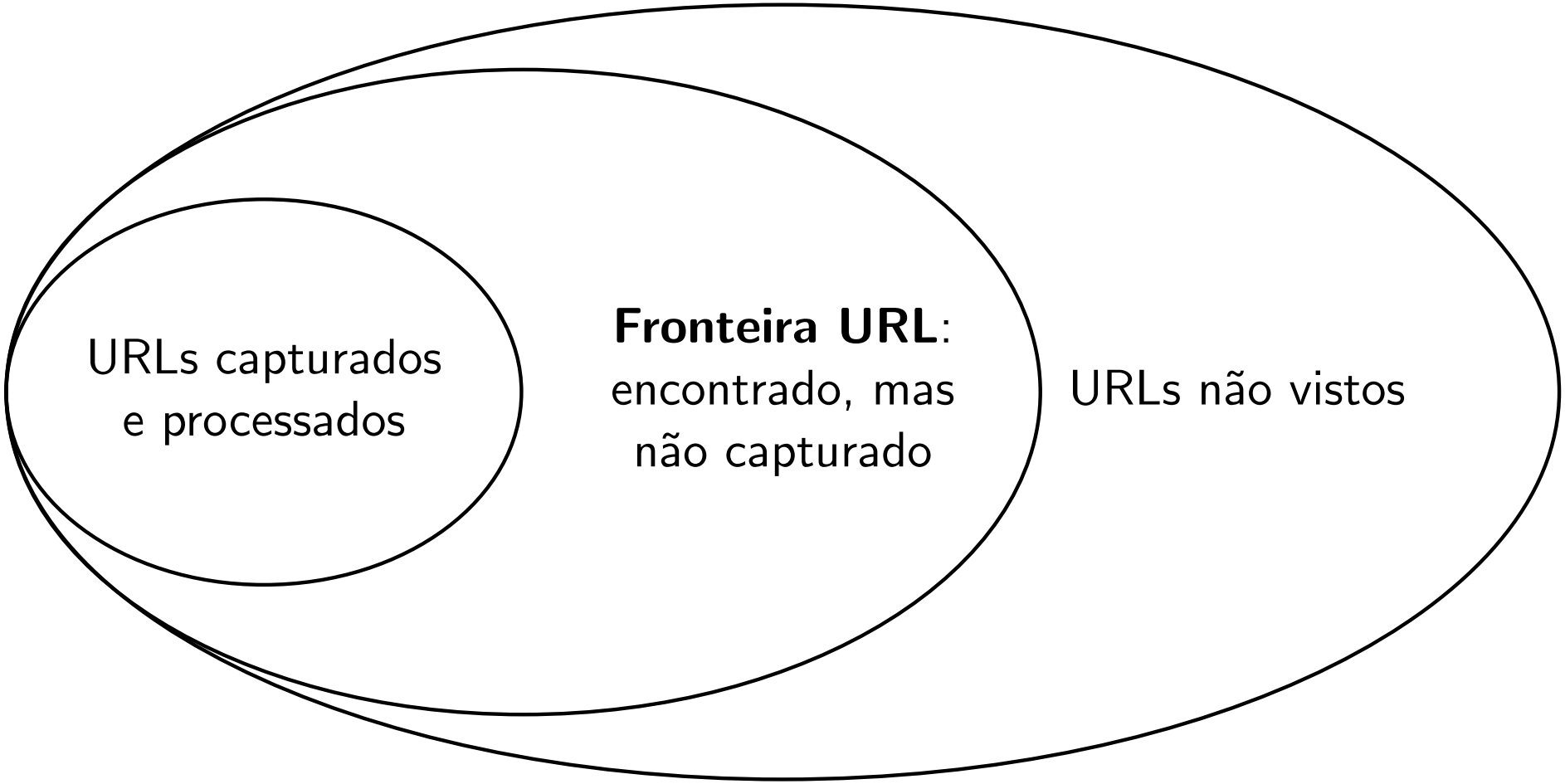
Fronteira URL



Fronteira URL



Fronteira URL



Fronteira URL

- Fronteira URL é uma estrutura de dados que guarda e gerencia URLs encontrados, mas ainda não capturados

Fronteira URL

- Fronteira URL é uma estrutura de dados que guarda e gerencia URLs encontrados, mas ainda não capturados
- Pode incluir múltiplas páginas do mesmo servidor

Fronteira URL

- Fronteira URL é uma estrutura de dados que guarda e gerencia URLs encontrados, mas ainda não capturados
- Pode incluir múltiplas páginas do mesmo servidor
- Deve evitar captura de todas ao mesmo tempo

Fronteira URL

- Fronteira URL é uma estrutura de dados que guarda e gerencia URLs encontrados, mas ainda não capturados
- Pode incluir múltiplas páginas do mesmo servidor
- Deve evitar captura de todas ao mesmo tempo
- Deve manter processos de crawling ocupados

Passos em crawling

- Escolher uma URL da fronteira

Passos em crawling

- Escolher uma URL da fronteira
- Capturar o documento na URL

Passos em crawling

- Escolher uma URL da fronteira
- Capturar o documento na URL
- Se documento não for conhecido fazer:

Passos em crawling

- Escolher uma URL da fronteira
- Capturar o documento na URL
- Se documento não for conhecido fazer:
  - Indexar o documento

Passos em crawling

- Escolher uma URL da fronteira
- Capturar o documento na URL
- Se documento não for conhecido fazer:
  - Indexar o documento
  - Extrair URLs do documento

Passos em crawling

- Escolher uma URL da fronteira
- Capturar o documento na URL
- Se documento não for conhecido fazer:
  - Indexar o documento
  - Extrair URLs do documento
  - Para cada URL extraída:

Passos em crawling

- Escolher uma URL da fronteira
- Capturar o documento na URL
- Se documento não for conhecido fazer:
  - Indexar o documento
  - Extrair URLs do documento
  - Para cada URL extraída:
    - Realizar testes de qualidade.

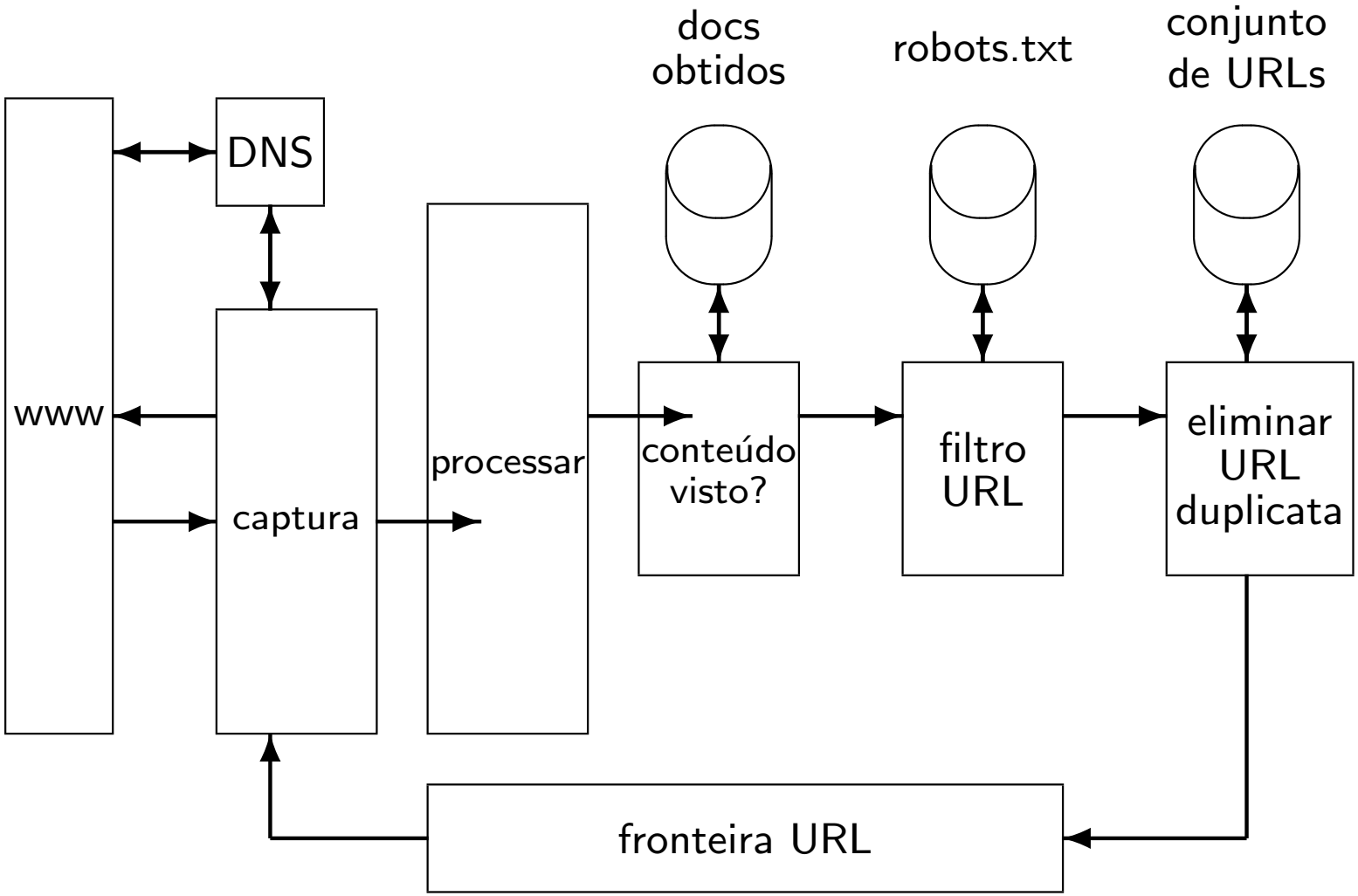
Passos em crawling

- Escolher uma URL da fronteira
- Capturar o documento na URL
- Se documento não for conhecido fazer:
  - Indexar o documento
  - Extrair URLs do documento
  - Para cada URL extraída:
    - Realizar testes de qualidade.
      Exemplo: é spam? Se sim: ignorar.

Passos em crawling

- Escolher uma URL da fronteira
- Capturar o documento na URL
- Se documento não for conhecido fazer:
  - Indexar o documento
  - Extrair URLs do documento
  - Para cada URL extraída:
    - Realizar testes de qualidade.
      Exemplo: é spam? Se sim: ignorar.
    - Já na fronteira? Se sim: pular.

Arquitetura básica



Normalização de URLs

- Alguns URLs extraídos de documentos são relativos
- Em `http://www.ufu.br` achamos: "`/pagina/sobre-ufu`"
  - O que é o mesmo que:
    `http://www.ufu.br/pagina/sobre-ufu`
- Durante processamento, devemos normalizar todas as URLs relativas

Conteúdo visto

- Para cada página obtida: verificar se conteúdo já está no índice
  - Usar "impressões digitais" do documento com funções hash
- Pular documentos cujo conteúdo já foi indexado

Distribuindo o crawler

- Rodar múltiplos processos de crawling em computadores distintos

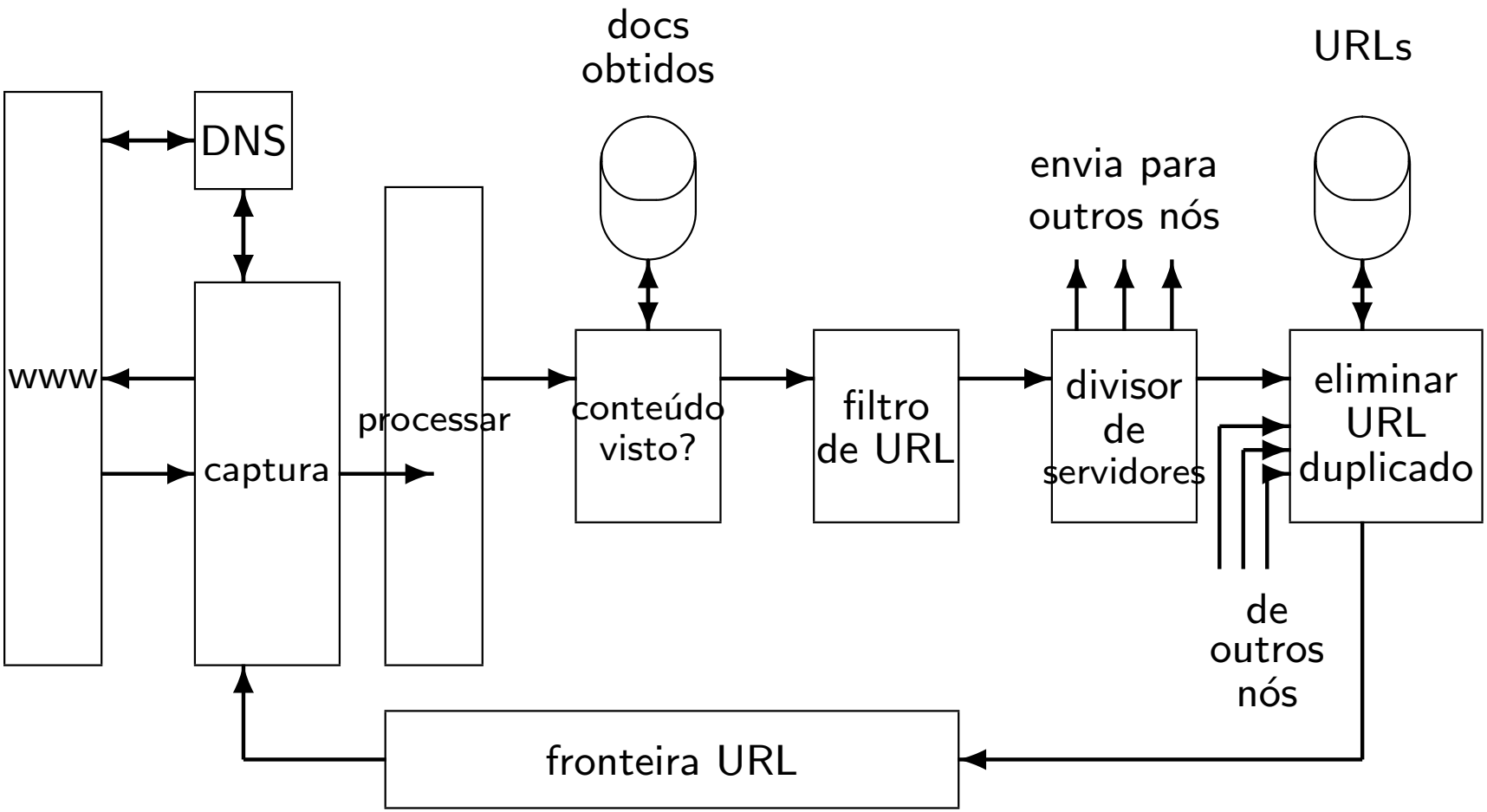
Distribuindo o crawler

- Rodar múltiplos processos de crawling em computadores distintos
  - Normalmente em computadores geograficamente espalhados

Distribuindo o crawler

- Rodar múltiplos processos de crawling em computadores distintos
  - Normalmente em computadores geograficamente espalhados
- Dividir servidores-alvo entre esses computadores

Crawler distribuído



Fronteira URL: duas considerações principais

- Cortesia: não sobrecarregar servidores

Fronteira URL: duas considerações principais

- Cortesia: não sobrecarregar servidores
  - E.g., esperar um tempo entre requisições sucessivas para um mesmo servidor

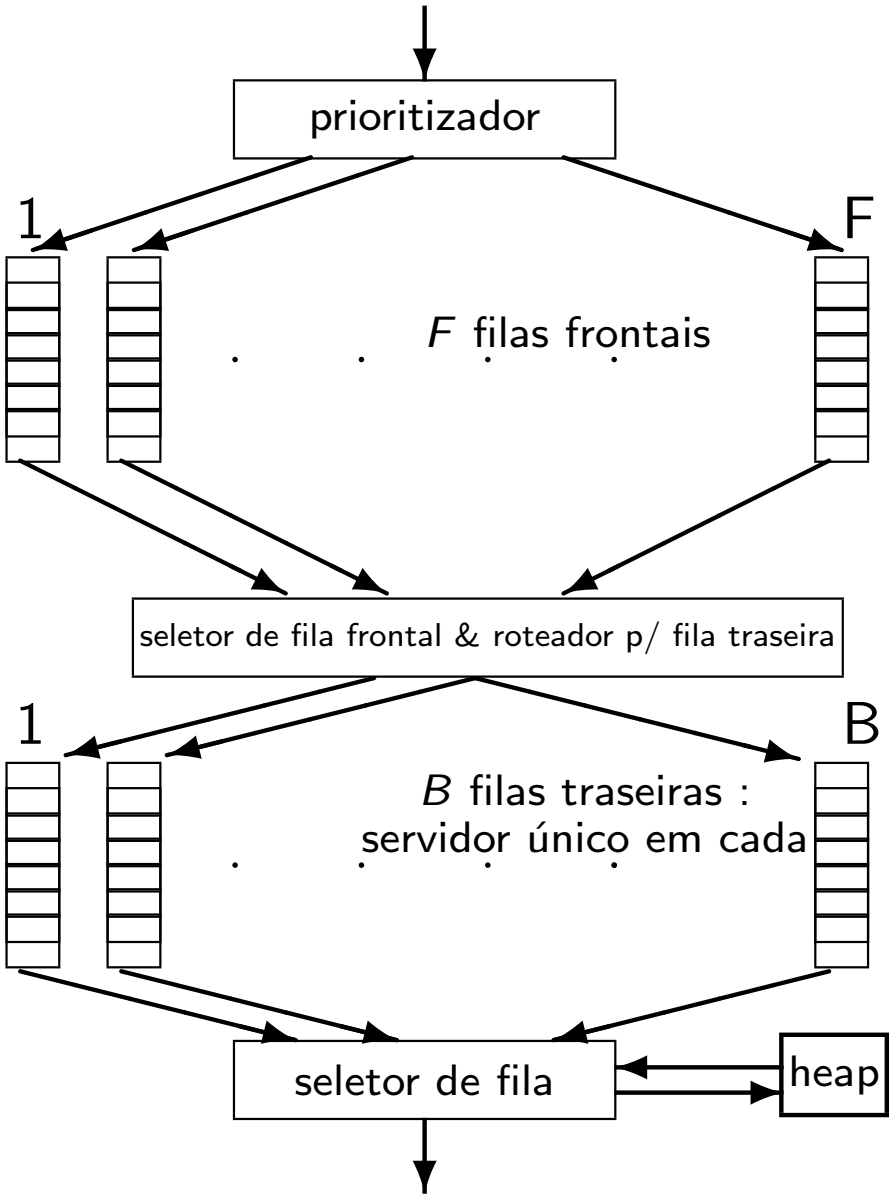
Fronteira URL: duas considerações principais

- Cortesia: não sobrecarregar servidores
  - E.g., esperar um tempo entre requisições sucessivas para um mesmo servidor
- Recência: capturar algumas páginas (e.g., sites de notícias) com maior frequência que outras

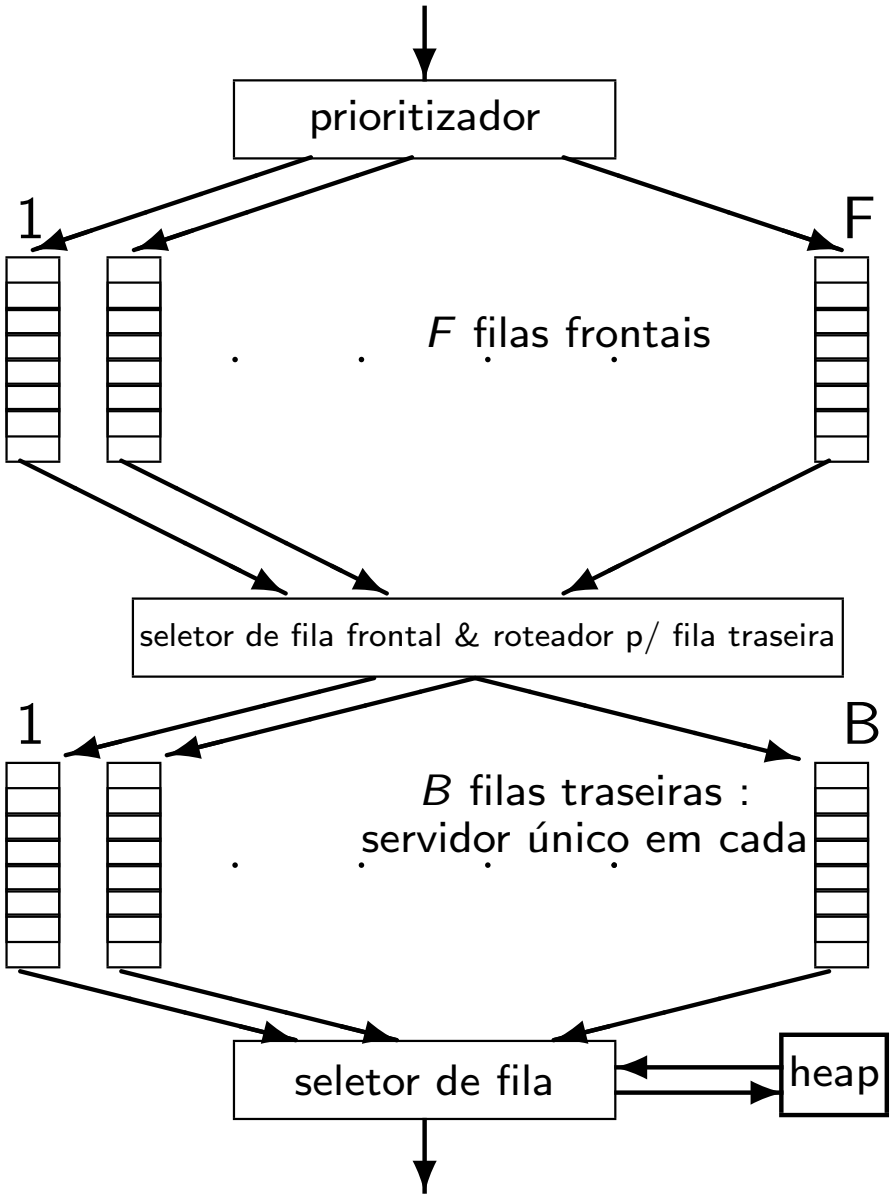
Fronteira URL: duas considerações principais

- Cortesia: não sobrecarregar servidores
  - E.g., esperar um tempo entre requisições sucessivas para um mesmo servidor
- Recência: capturar algumas páginas (e.g., sites de notícias) com maior frequência que outras
- Problema complexo: não é possível usar fila de prioridade simples

Fronteira URL de Mercator

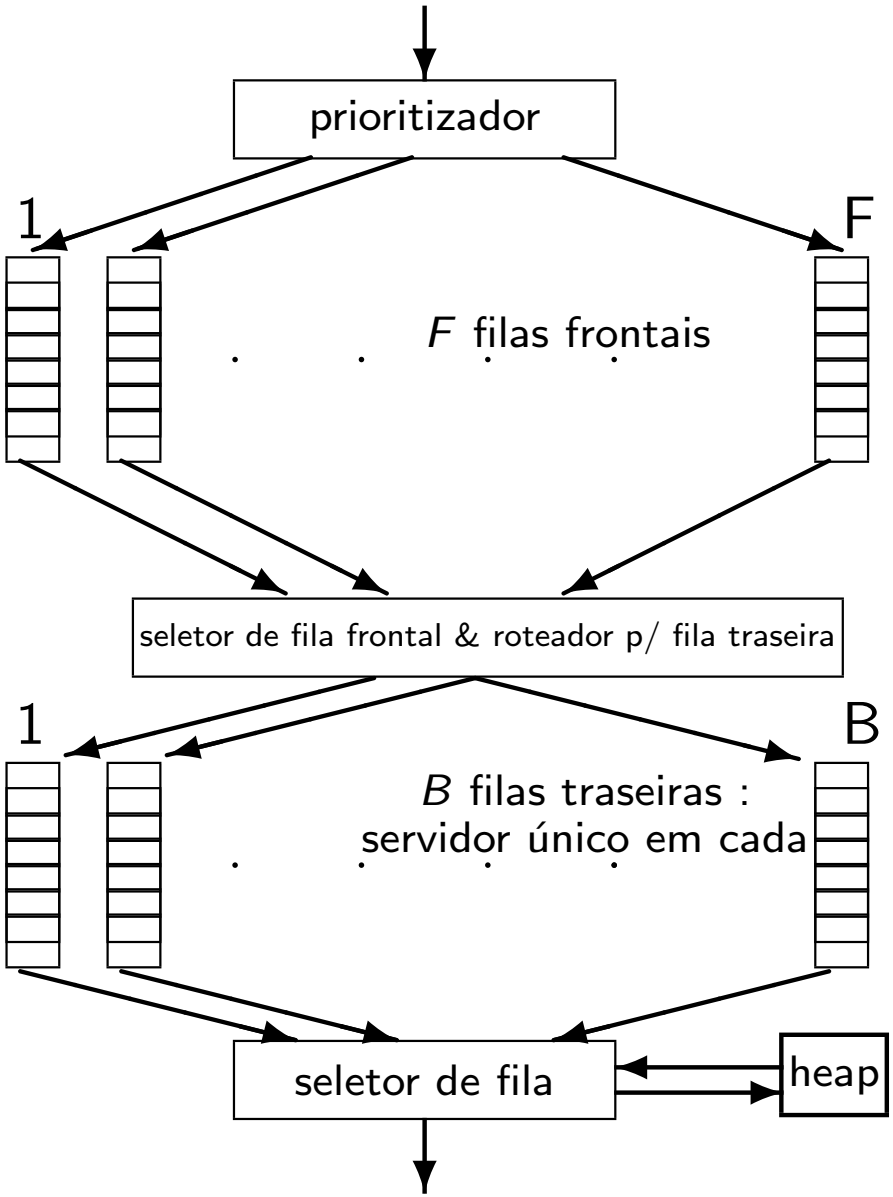


Fronteira URL de Mercator



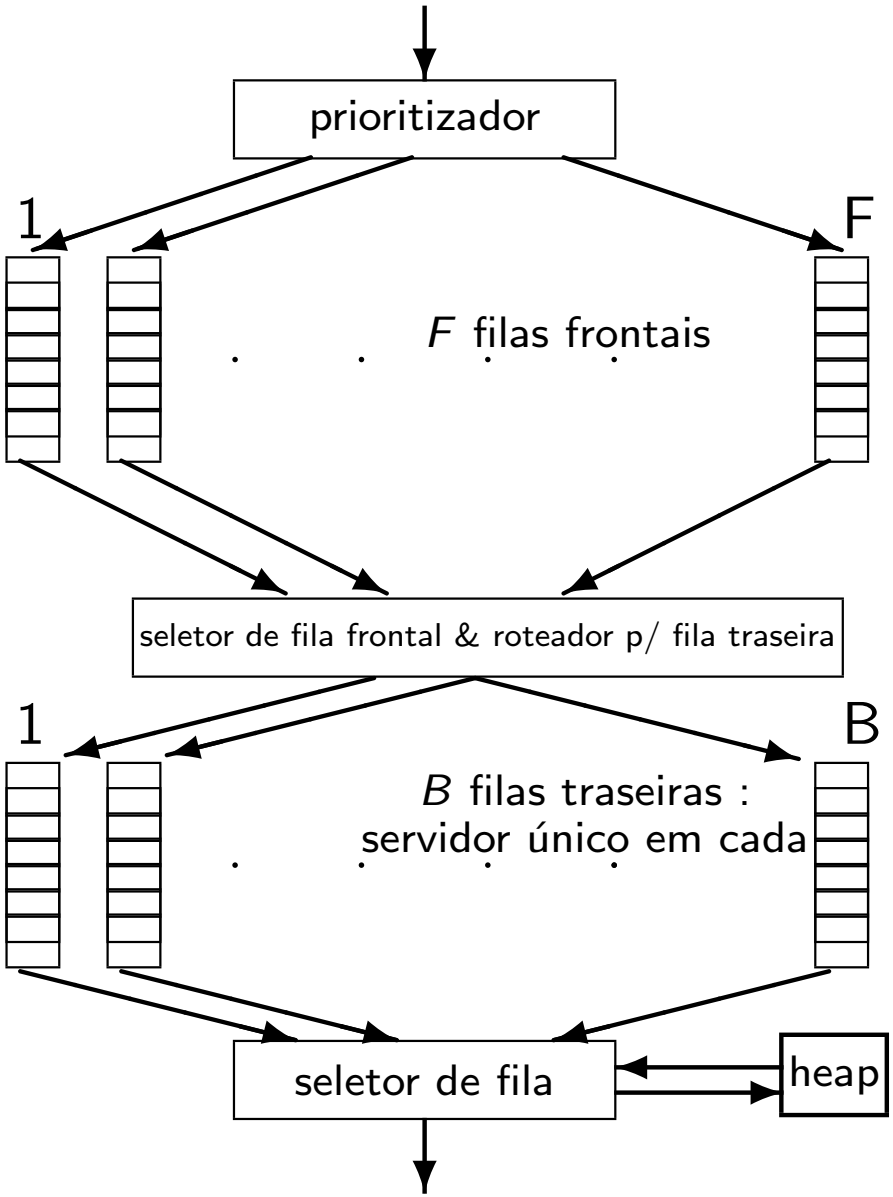
- Fluxo de URLs de entrada do topo para a fronteira

Fronteira URL de Mercator



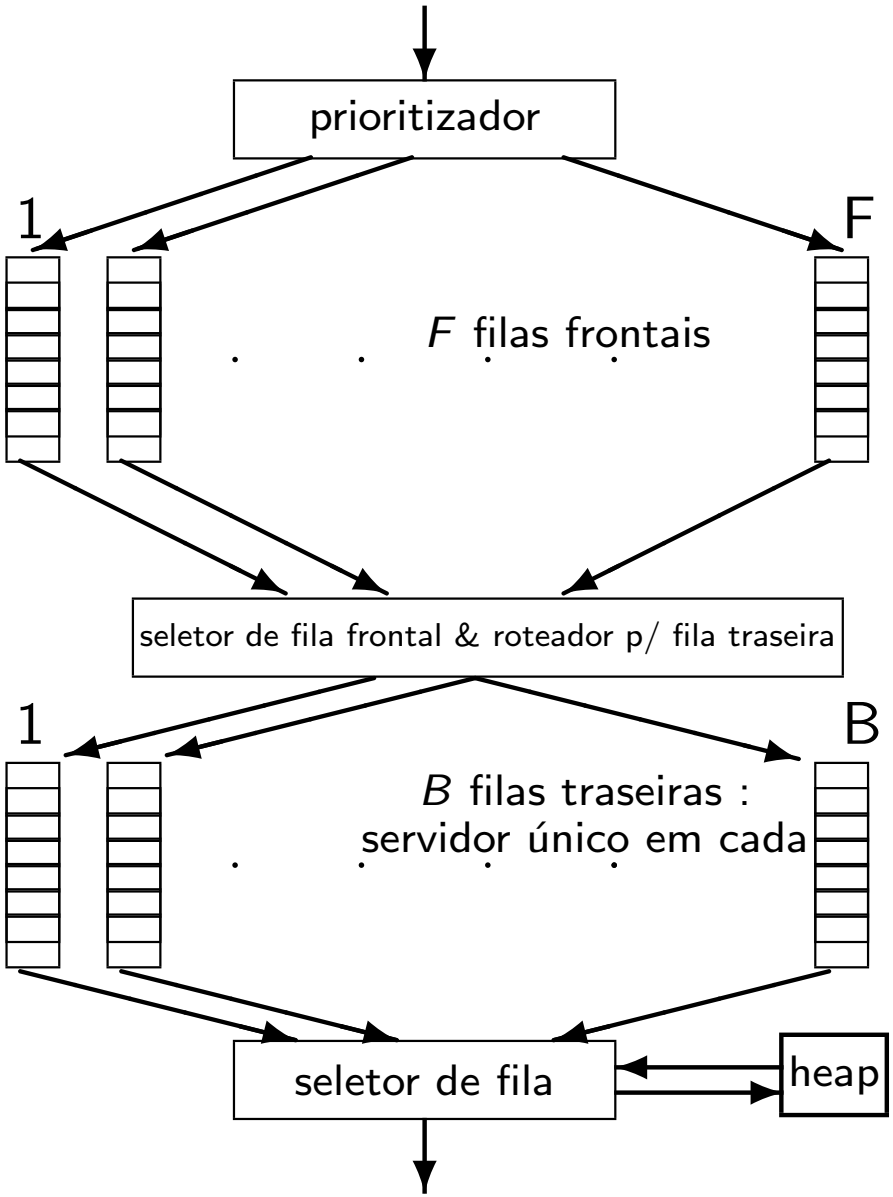
- Fluxo de URLs de entrada do topo para a fronteira
- Filas de frente gerenciam recência

Fronteira URL de Mercator



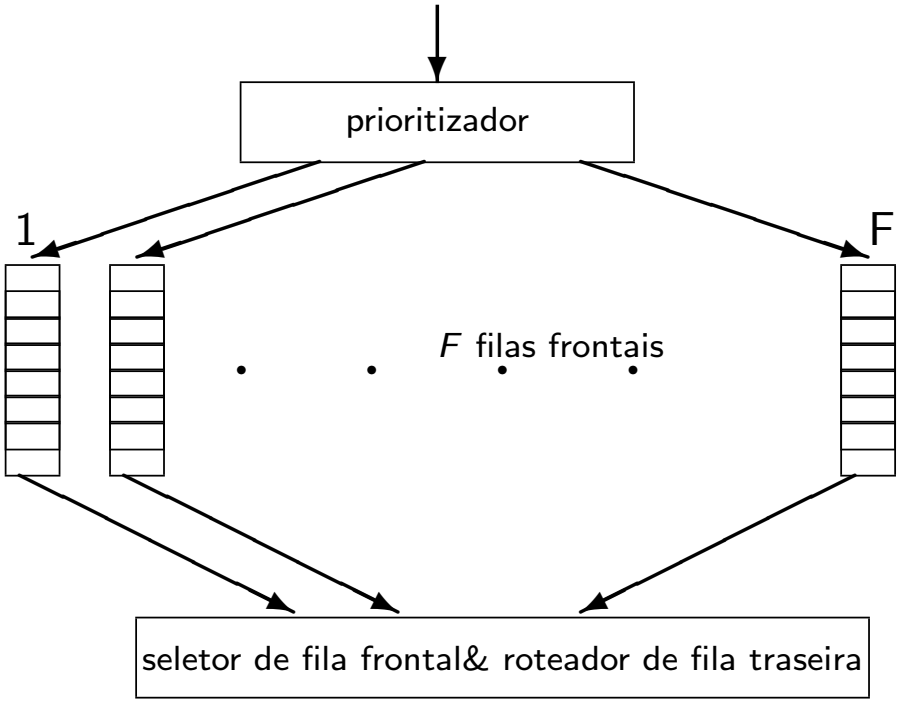
- Fluxo de URLs de entrada do topo para a fronteira
- Filas de frente gerenciam recência
- Filas traseiras garantem cortesia

Fronteira URL de Mercator

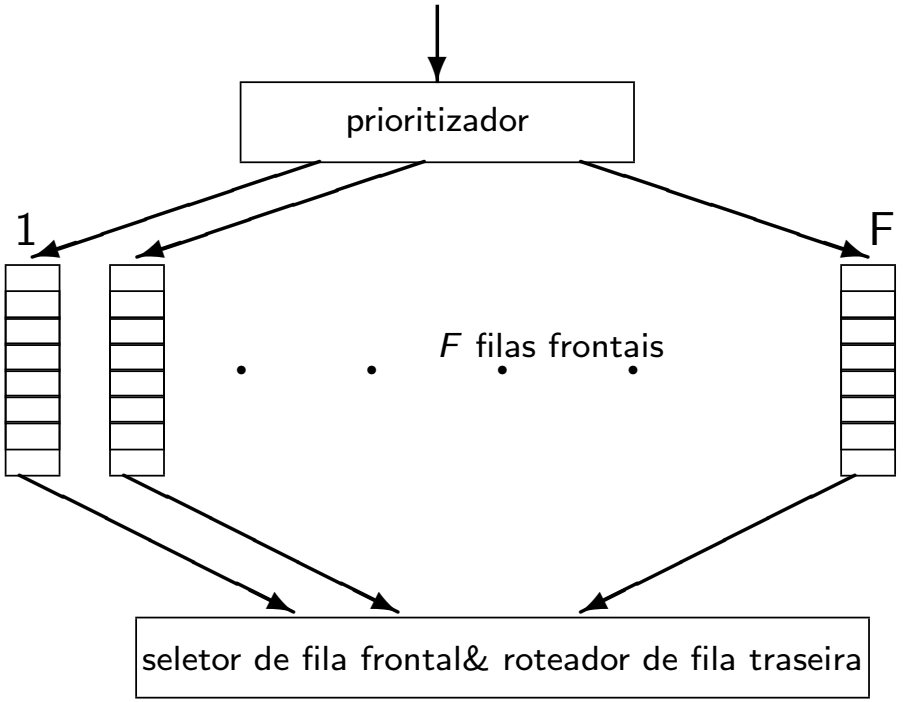


- Fluxo de URLs de entrada do topo para a fronteira
- Filas de frente gerenciam recência
- Filas traseiras garantem cortesia

Fronteira URL de Mercator: filas frontais

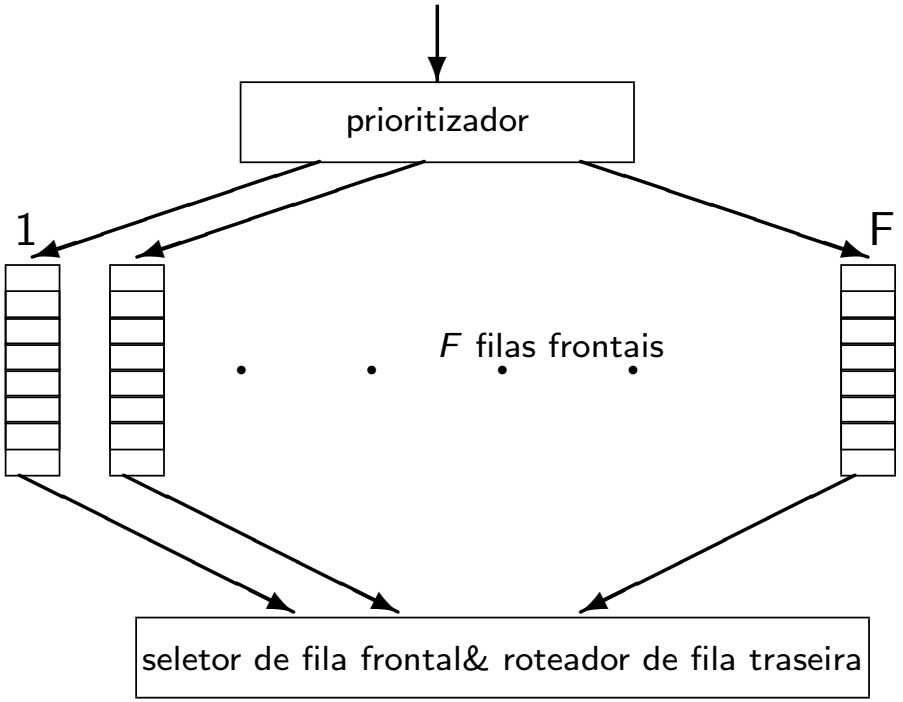


Fronteira URL de Mercator: filas frontais



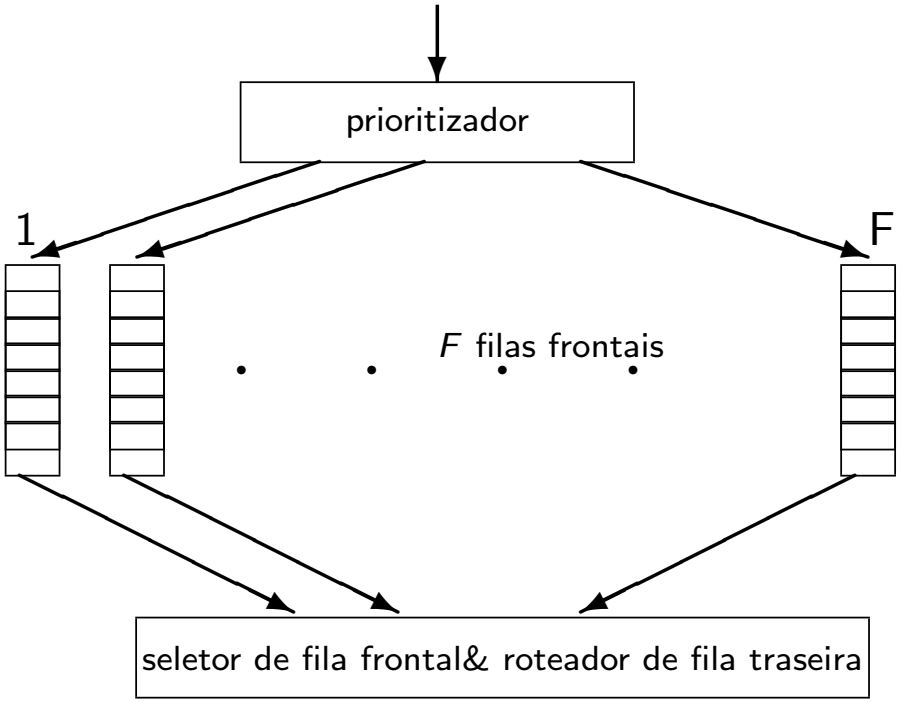
- Prioritizador atribui para URL um número inteiro de prioridade entre 1 e F.
- Então coloca URL à fila correspondente

Fronteira URL de Mercator: filas frontais



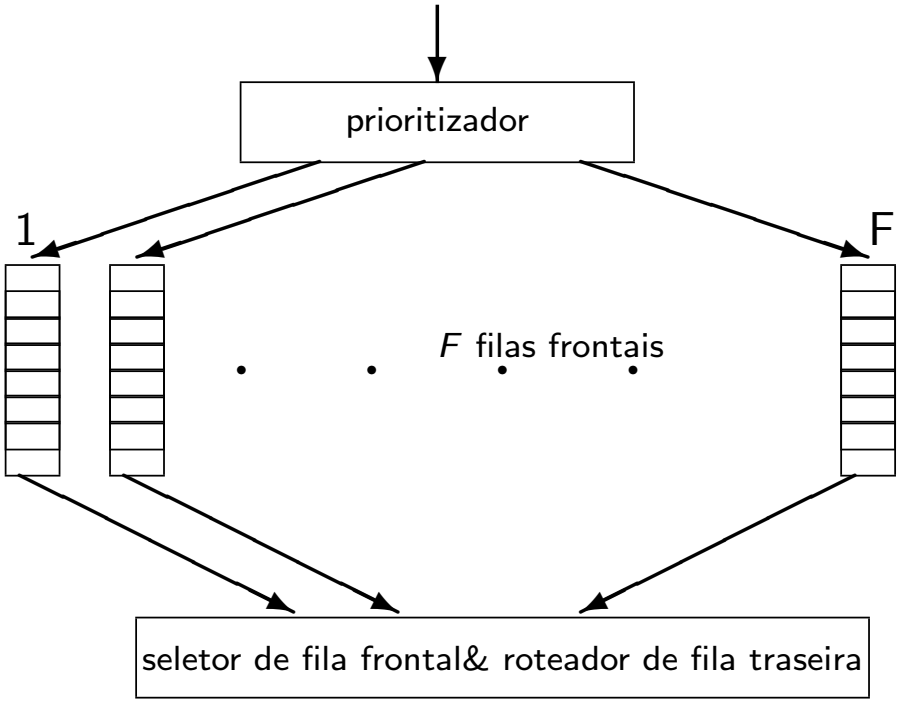
- Prioritizador atribui para URL um número inteiro de prioridade entre 1 e F.
- Então coloca URL à fila correspondente
- Heurística para atribuir prioridade: taxa de recência, PageRank etc

Fronteira URL de Mercator: filas frontais



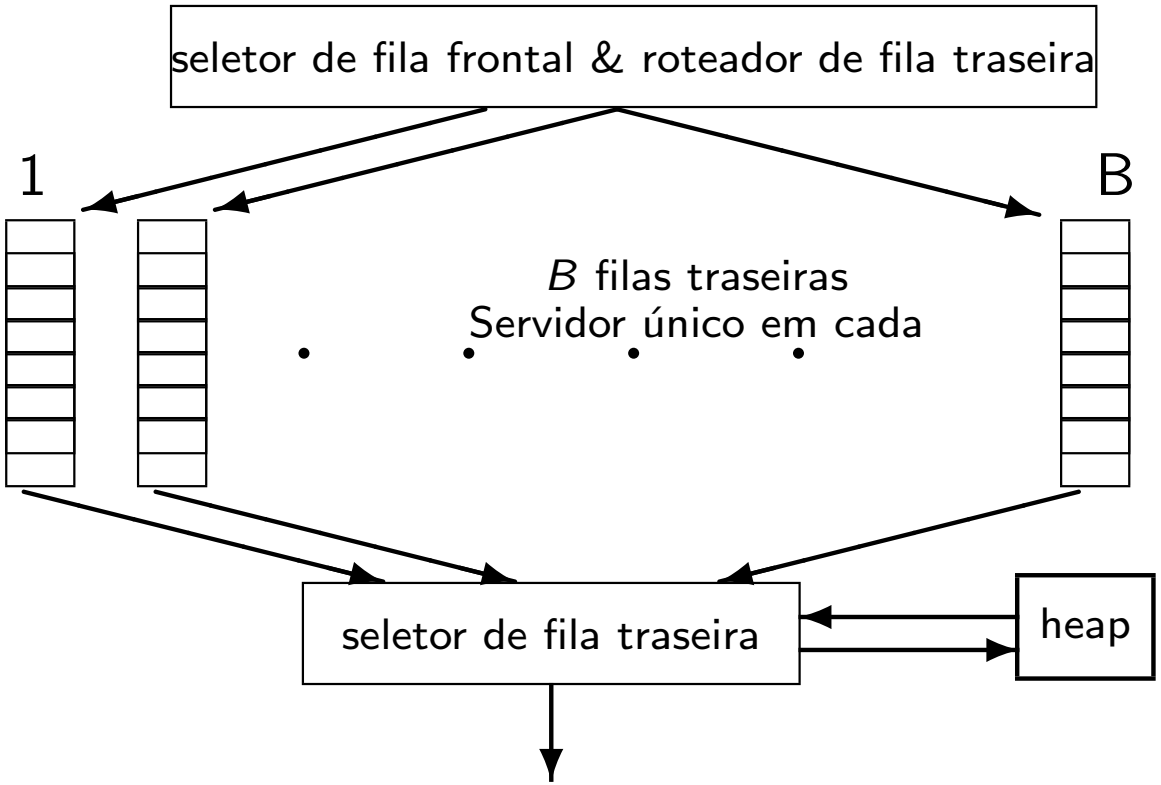
- Prioritizador atribui para URL um número inteiro de prioridade entre 1 e F.
- Então coloca URL à fila correspondente
- Heurística para atribuir prioridade: taxa de recência, PageRank etc

Fronteira URL de Mercator: filas frontais

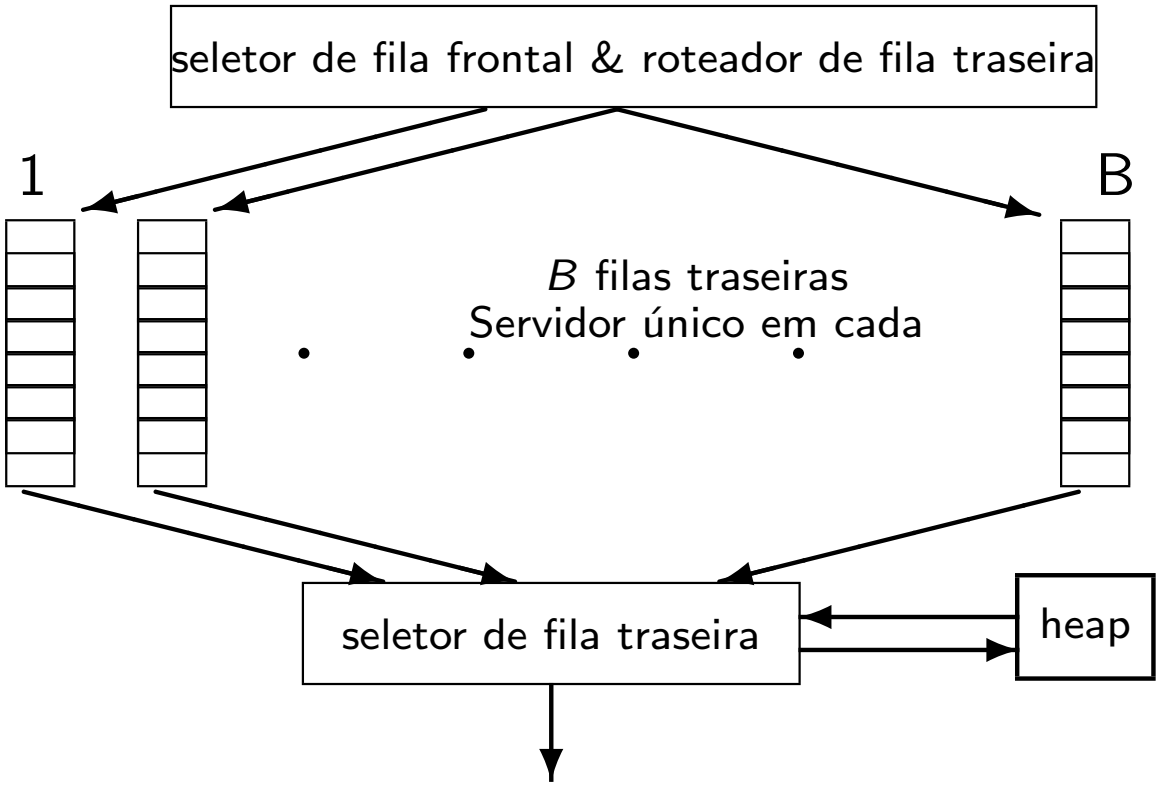


- Seleção das filas de frente é iniciada por filas traseiras
- Escolher uma fila de frente a partir do qual seleciona-se próximo URL: rodízio, aleatoriamente, ou variante mais sofisticada
- Mas com uma tendência a favor de filas de frente de alta prioridade

Fronteira de URL de Mercator: filas traseiras

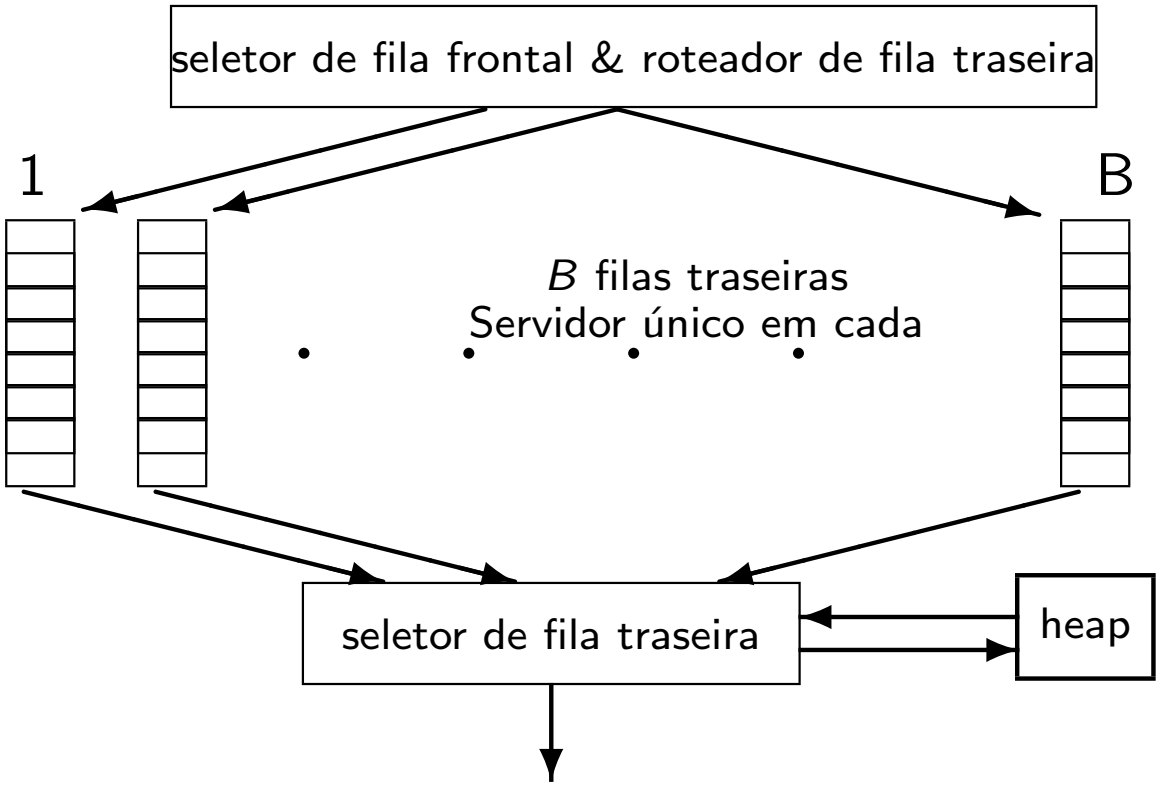


Fronteira de URL de Mercator: filas traseiras



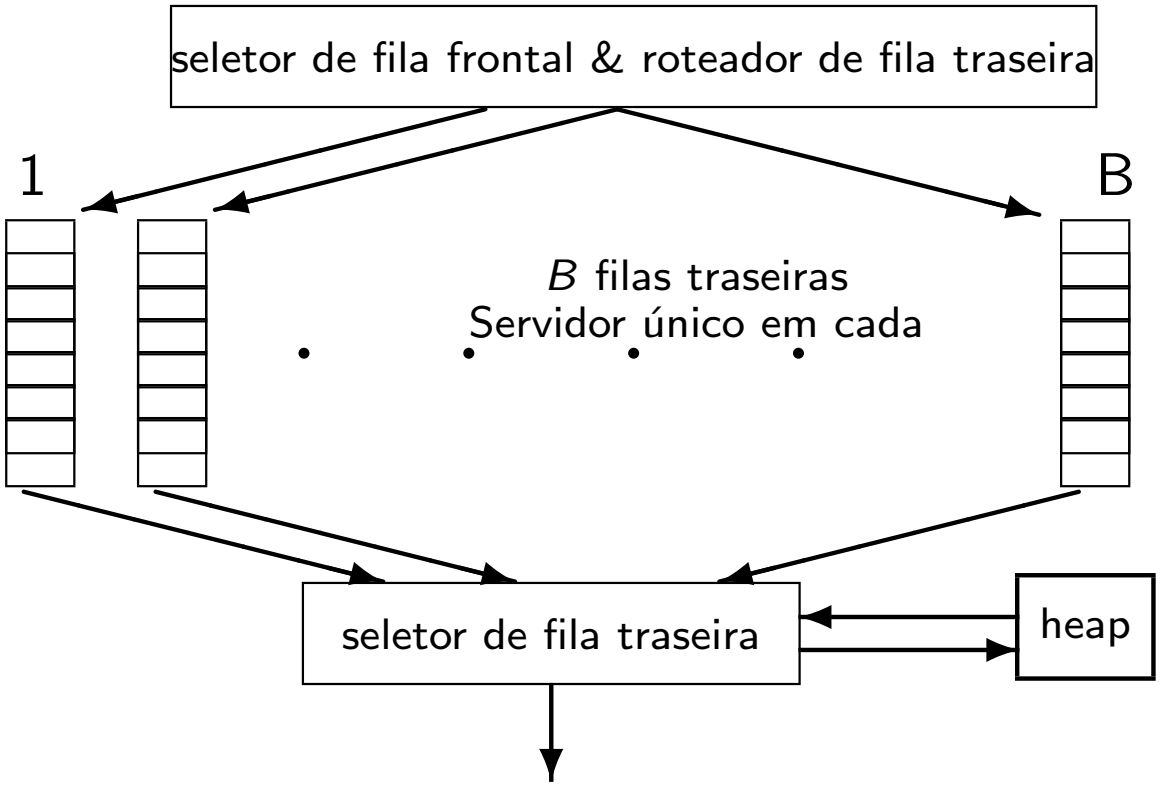
- Invariante 1. Cada fila traseira é mantida com páginas enquanto a captura está em progresso
- Invariante 2. Cada fila traseira somente contém URLs de um único servidor
- Mantém uma tabela de servidores para filas traseiras

Fronteira de URL de Mercator: filas traseiras



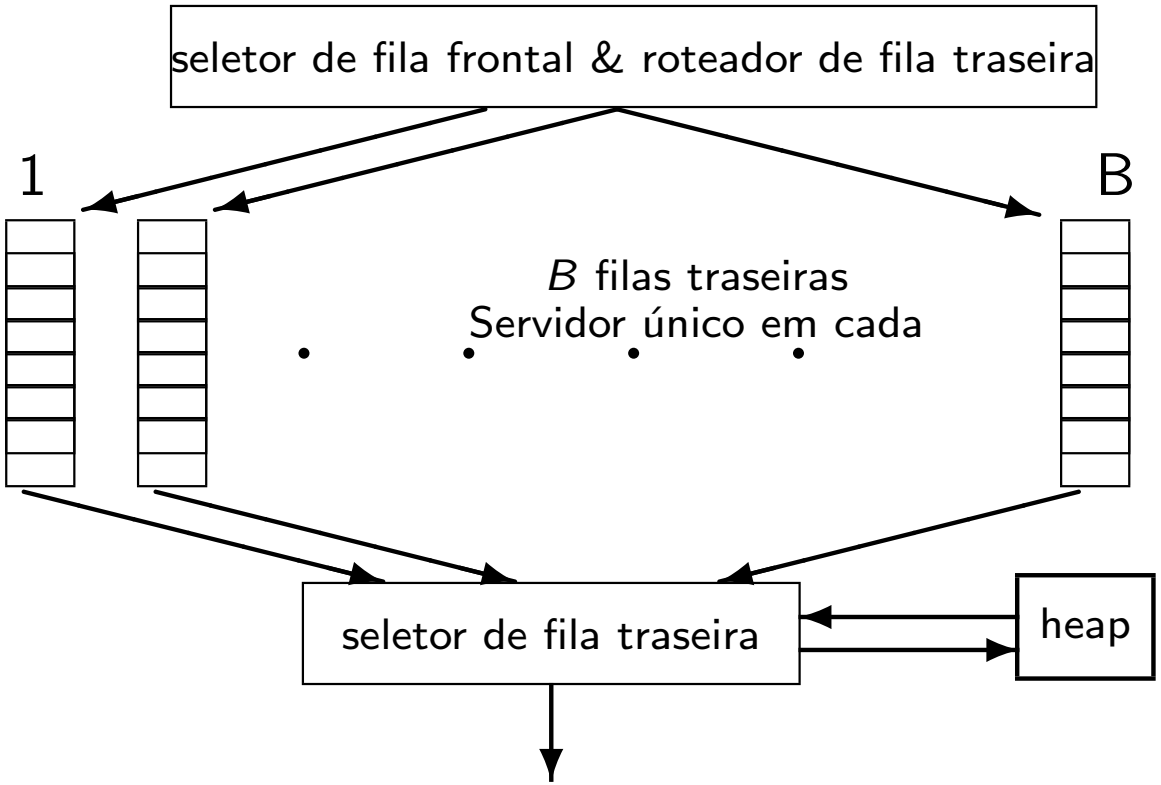
- Na heap:
- Uma entrada para cada fila traseira
- A entrada é o tempo mais próximo t_e em que o servidor correspondente para a fila traseira ser atingida novamente
- O tempo mais próximo t_e é determinado por (i) último acesso para aquele servidor e (ii) heurística do intervalo de tempo

Fronteira de URL de Mercator: filas traseiras



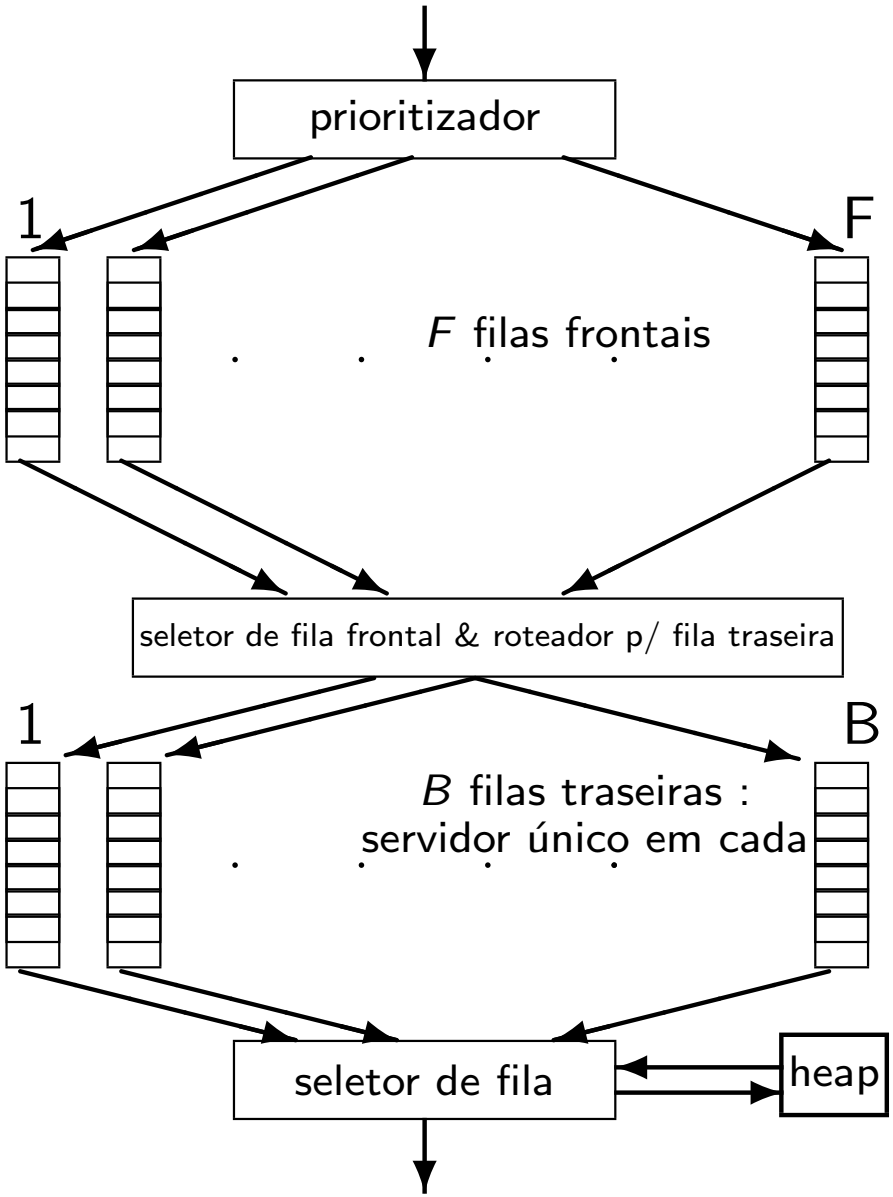
- Como o capturador interage com filas traseiras:
- Repetir (i) extrair raiz atual q da heap (q é uma fila traseira)
- e (ii) capturar URL u na cabeça de q . . .
- . . . até esvaziarmos q

Fronteira de URL de Mercator: filas traseiras



- Quando esvaziarmos uma fila traseira q:
- Repetir (i) tirar URLs u das filas da frente e (ii) adicionar u para suas correspontes filas traseiras . . .
- . . . até pegarmos um u cujo servidor não temos uma fila traseira
- Então colocamos u em q criamos um entrada da heap para ele

Fronteira URL de Mercator



- Fluxo de URLs de entrada do topo para a fronteira
- Filas de frente gerenciam recência
- Filas traseiras garantem cortesia

Armadilha para "aranha"

- Servidor malicioso que gera uma sequência infinita de páginas linkadas
- Armadilhas sofisticadas geram páginas que não são facilmente identificadas como dinâmicas
- Verificação da profundidade de URLs – fazer poucos níveis por vez
- Uso de `sitemap.xml` onde os URLs mais importantes são descritos
  - `http://www.google.com/sitemap.xml`
- Uso do robots.txt para afetar crawlers de spammers
- Uso de links partindo de páginas com page rank pré-calculado
- Uso de medidas de qualidade: textos estão legíveis?

Resumo

- Um crawler simples
- Crawling na web
- robots.txt
- Fronteira URL
- Arquitetura distribuída de crawling
- Fronteira URL de Mercator
  - `Mercator: A Scalable, Extensible Web Crawler`
- Armadilhas