Webspider

Dr. Christian Herta

June 11, 2009

Referenz-Architektur: Mercator

- Referenz-Architektur: Webcrawler Mercator [2]
- modular und Komponenten-basiert, so wird die Anforderung *extensible* erfüllt.
- (*separation of concern*): unterschiedliche Aufgaben/Schritte werden von unterschiedlichen Modulen ausgeführt

Schritte beim Spidern

1. Nehme die "nächste" URL aus der *Frontier*

Schritte beim Spidern

1. Nehme die "nächste" URL aus der *Frontier*
2. Hole (*Fetch*) das Dokument zu der URL aus dem Netz

Schritte beim Spidern

1. Nehme die "nächste" URL aus der *Frontier*
2. Hole (*Fetch*) das Dokument zu der URL aus dem Netz
3. Überprüfe, ob der Inhalt des Dokuments schon "gesehen" wurde; falls ja, überspringe die folgenden Schritte

Schritte beim Spidern

1. Nehme die "nächste" URL aus der *Frontier*
2. Hole (*Fetch*) das Dokument zu der URL aus dem Netz
3. Überprüfe, ob der Inhalt des Dokuments schon "gesehen" wurde; falls ja, überspringe die folgenden Schritte
4. Speichere das Dokument zur weiteren Verarbeitung

Schritte beim Spidern

1. Nehme die "nächste" URL aus der *Frontier*
2. Hole (*Fetch*) das Dokument zu der URL aus dem Netz
3. Überprüfe, ob der Inhalt des Dokuments schon "gesehen" wurde; falls ja, überspringe die folgenden Schritte
4. Speichere das Dokument zur weiteren Verarbeitung
5. Parse das Dokument und extrahiere alle URLs

Schritte beim Spidern

1. Nehme die "nächste" URL aus der *Frontier*
2. Hole (*Fetch*) das Dokument zu der URL aus dem Netz
3. Überprüfe, ob der Inhalt des Dokuments schon "gesehen" wurde; falls ja, überspringe die folgenden Schritte
4. Speichere das Dokument zur weiteren Verarbeitung
5. Parse das Dokument und extrahiere alle URLs
6. für jede extrahierte URL:

Schritte beim Spidern

1. Nehme die "nächste" URL aus der *Frontier*
2. Hole (*Fetch*) das Dokument zu der URL aus dem Netz
3. Überprüfe, ob der Inhalt des Dokuments schon "gesehen" wurde; falls ja, überspringe die folgenden Schritte
4. Speichere das Dokument zur weiteren Verarbeitung
5. Parse das Dokument und extrahiere alle URLs
6. für jede extrahierte URL:
   - Überprüfe die URL (nach Spam, regulären Ausdrücken etc.)

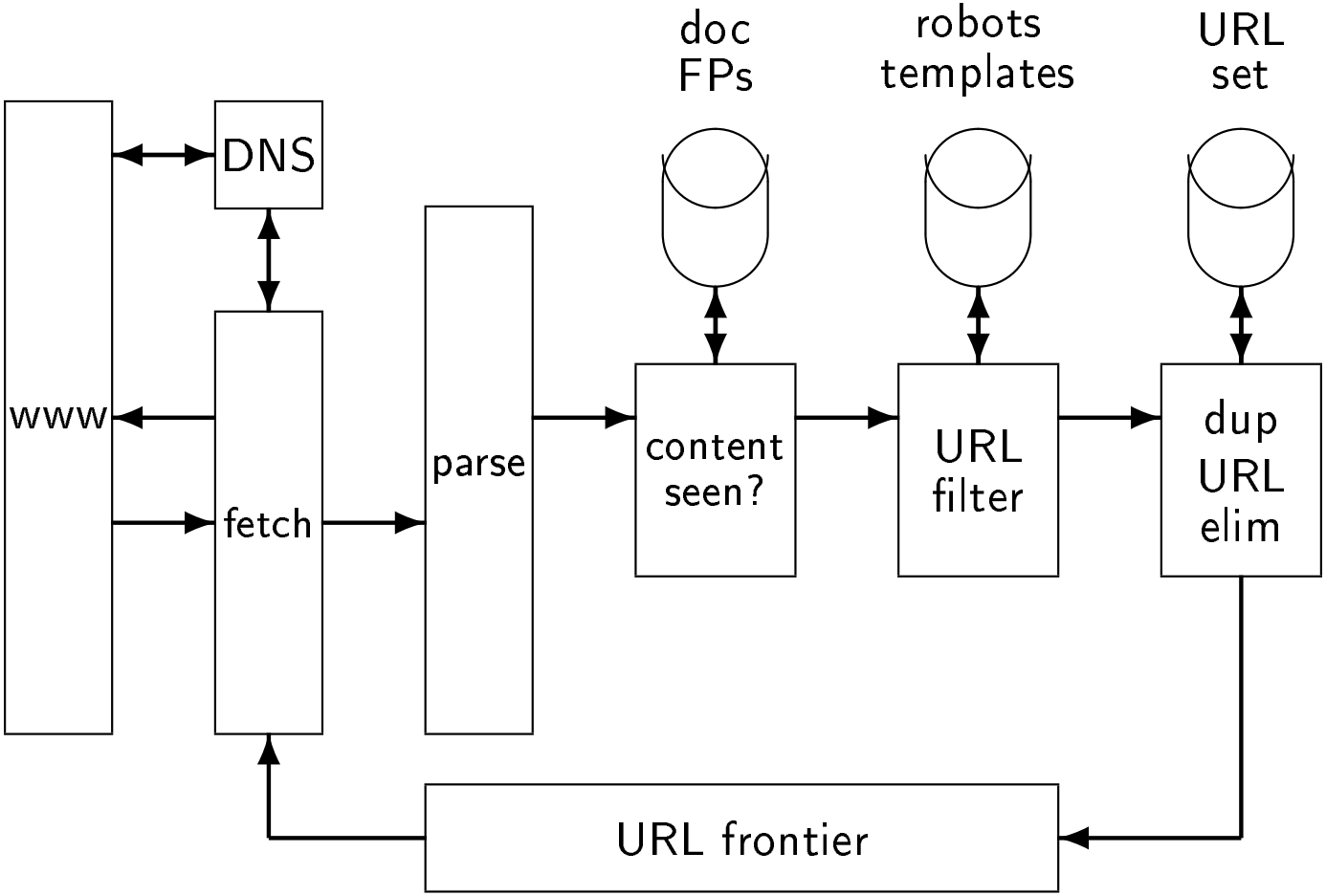
Schritte beim Spidern

1. Nehme die "nächste" URL aus der *Frontier*
2. Hole (*Fetch*) das Dokument zu der URL aus dem Netz
3. Überprüfe, ob der Inhalt des Dokuments schon "gesehen" wurde; falls ja, überspringe die folgenden Schritte
4. Speichere das Dokument zur weiteren Verarbeitung
5. Parse das Dokument und extrahiere alle URLs
6. für jede extrahierte URL:
   - Überprüfe die URL (nach Spam, regulären Ausdrücken etc.)
   - Ist die URL schon in der *Frontier*

Schritte beim Spidern

1. Nehme die "nächste" URL aus der *Frontier*
2. Hole (*Fetch*) das Dokument zu der URL aus dem Netz
3. Überprüfe, ob der Inhalt des Dokuments schon "gesehen" wurde; falls ja, überspringe die folgenden Schritte
4. Speichere das Dokument zur weiteren Verarbeitung
5. Parse das Dokument und extrahiere alle URLs
6. für jede extrahierte URL:
   - Überprüfe die URL (nach Spam, regulären Ausdrücken etc.)
   - Ist die URL schon in der *Frontier*
   - Gebe die URL der Frontier, wenn Kriterien (siehe oben) dies erlauben

Basic crawl architecture [1]



Komponenten eines Webcrawlers

- *DNS Resolver*: zugrundeliegende Netzwerk-Schicht `TCP` benötigt IP-Adresse - nicht Hostnamen
- *Fetcher*: Modul zum Holen der Seiten über das Netzwerk
- *Link Extractor*: Extrahieren der Link-URIs aus dem geholten Dokument
- *URL Filter*: Ausfilteren von URIs, z.B. nach `robots.txt` und regulären Ausdrücken
- *Duplicate Detector*: Duplikate erkennen
- *URL Frontier* zum Speichern der URL-Liste die heruntergeladen werden soll; Auswahl der URLs nach Priorität

URL Normalisierung

Definition: URL-Nomalisierung

Unter URL Normalisierung versteht man die Transformation einer URL in eine kanonische Form.

Beispiele für die Normalisierung:

- für relative URLs
  - z.B. auf der *Domain* `http://christianherta.de` gibt es relative Angabe `/home.html`
  - entspricht `http://christianherta.de/home.html`
- *Lowercasing*; folgende URLs sind äquivalent
  - `http://ChristianHerta.de`
  - `http://christianherta.de/index.html`
- Entfernen von *Session IDs* aus der URL

Duplikate *Content seen*

- Für jede Seite muss überprüft werden, ob der (fast) gleiche Inhalt nicht schon auf einer anderen Seite gefunden wurde
  - Volle Duplikate: Dokument-Fingerprint (Hash-Code)
  - Fast Duplikate (*near duplicates*): Shingles
- Überspringe Duplikate

Frontier Komponente

Begriff

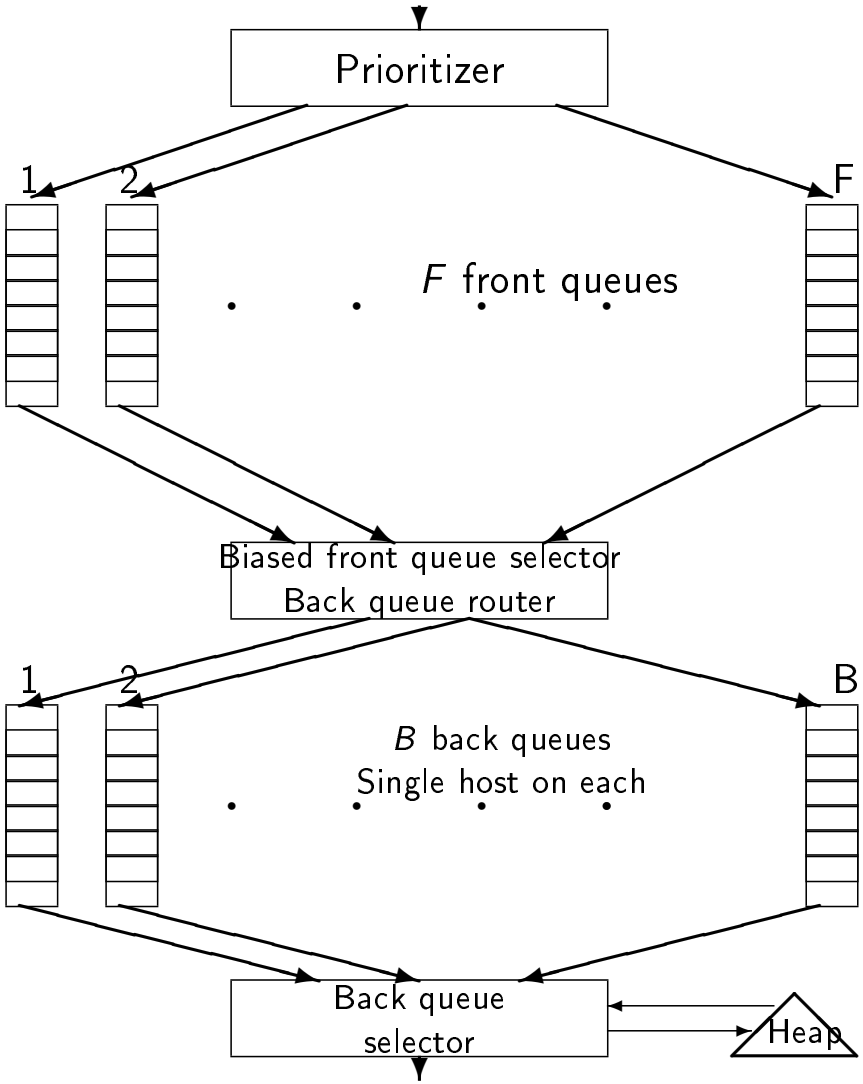
In der *Frontier* stehen die extrahierten URLs, die gecrawled werden sollen

- Synonym: (logische) *request queue*

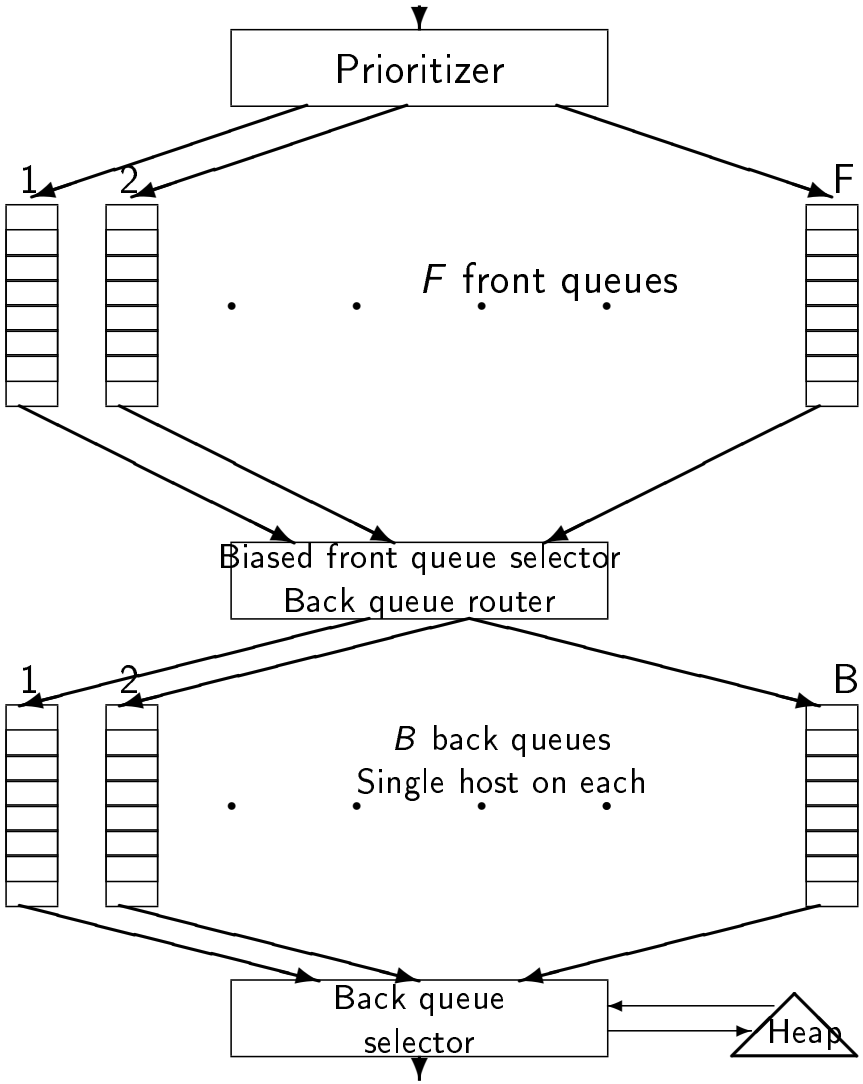
Aufgabe der *Frontier*

- Auswahl der URIs, die als nächstes gespidert werden sollen, unter Berücksichtigung verschiedener Kriterien:
  - *politeness policies*:
  - Auswahl der URIs nach Wichtigkeit, idealerweise bevorzugt Seiten mit höherer Qualität
  - bei incrementellem Spidern: gewährleisten der Aktualität der Web-Seiten - *update* Raten der Webseiten
- Auslasten aller Threads

Mercator URL Frontier [1][2]

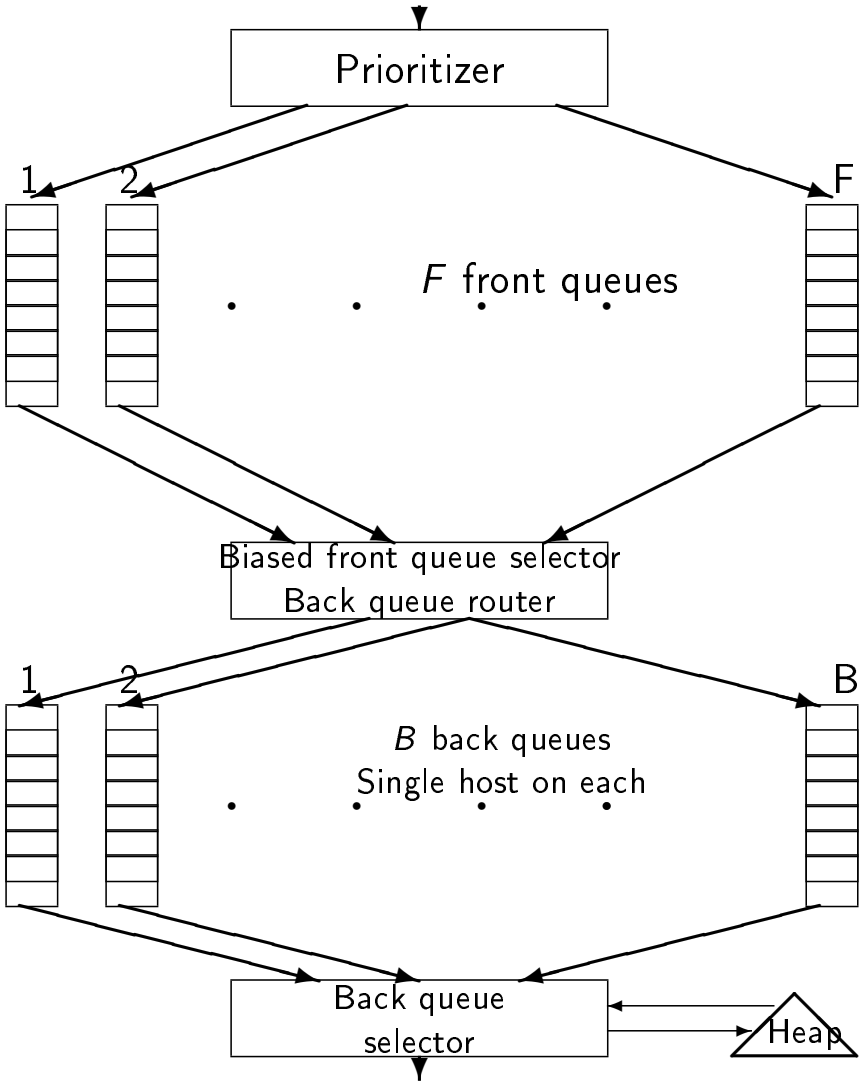


Mercator URL Frontier [1][2]



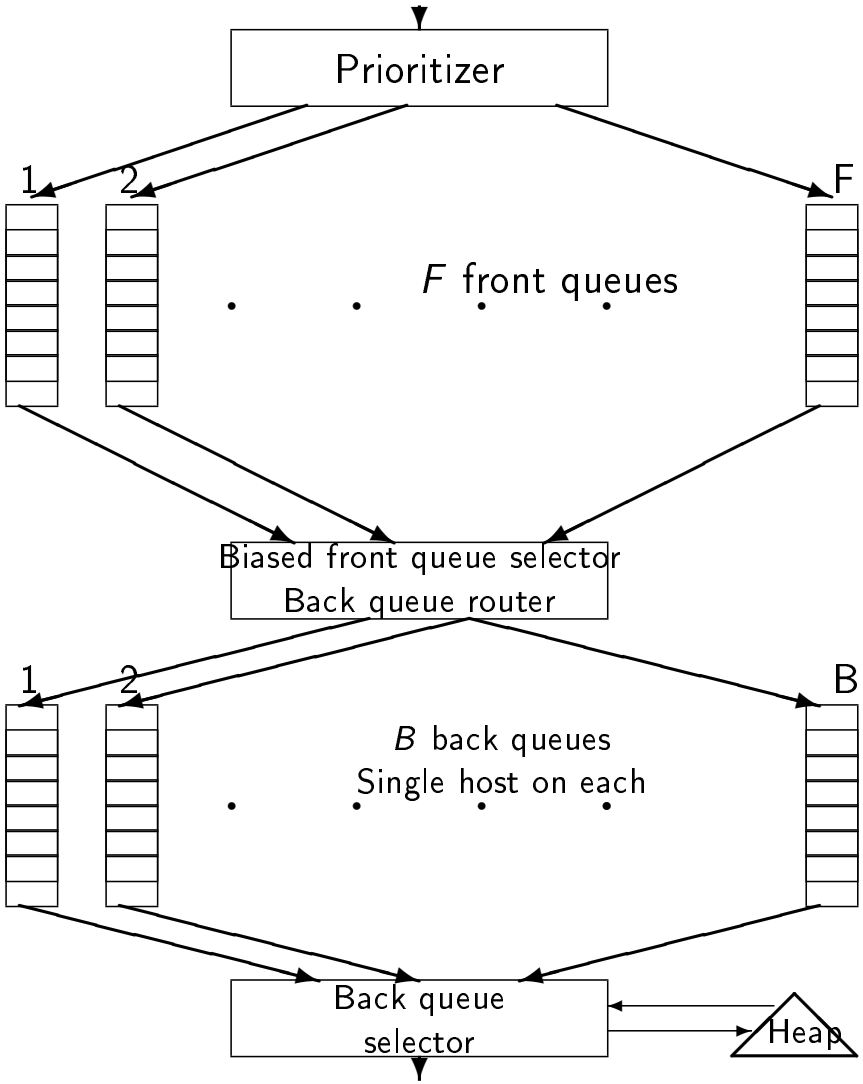
- URLs flow in from the top into the frontier.

Mercator URL Frontier [1][2]



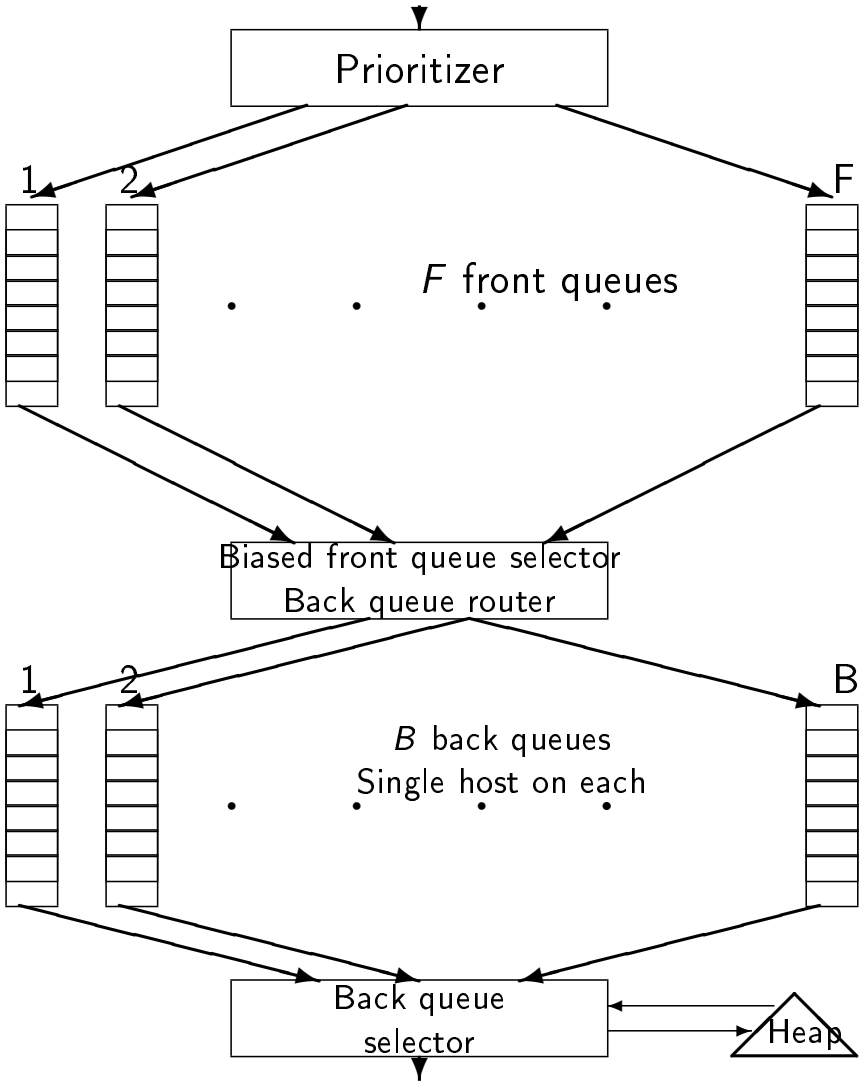
- URLs flow in from the top into the frontier.
- Front queues manage prioritization.

Mercator URL Frontier [1][2]



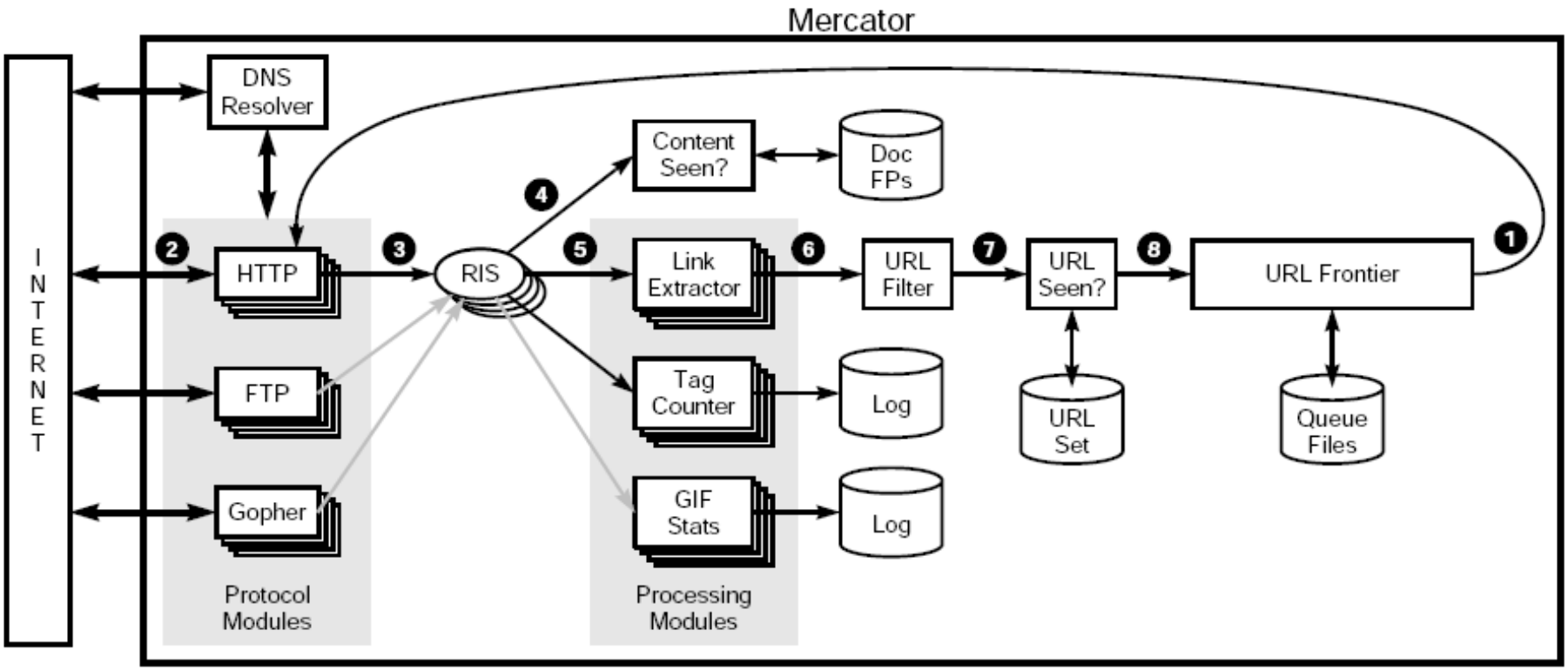
- URLs flow in from the top into the frontier.
- Front queues manage prioritization.
- Back queues enforce politness.

Mercator URL Frontier [1][2]



- URLs flow in from the top into the frontier.
- Front queues manage prioritization.
- Back queues enforce politness.
- Each queue is FIFO.

Architektur von Mercator [2]

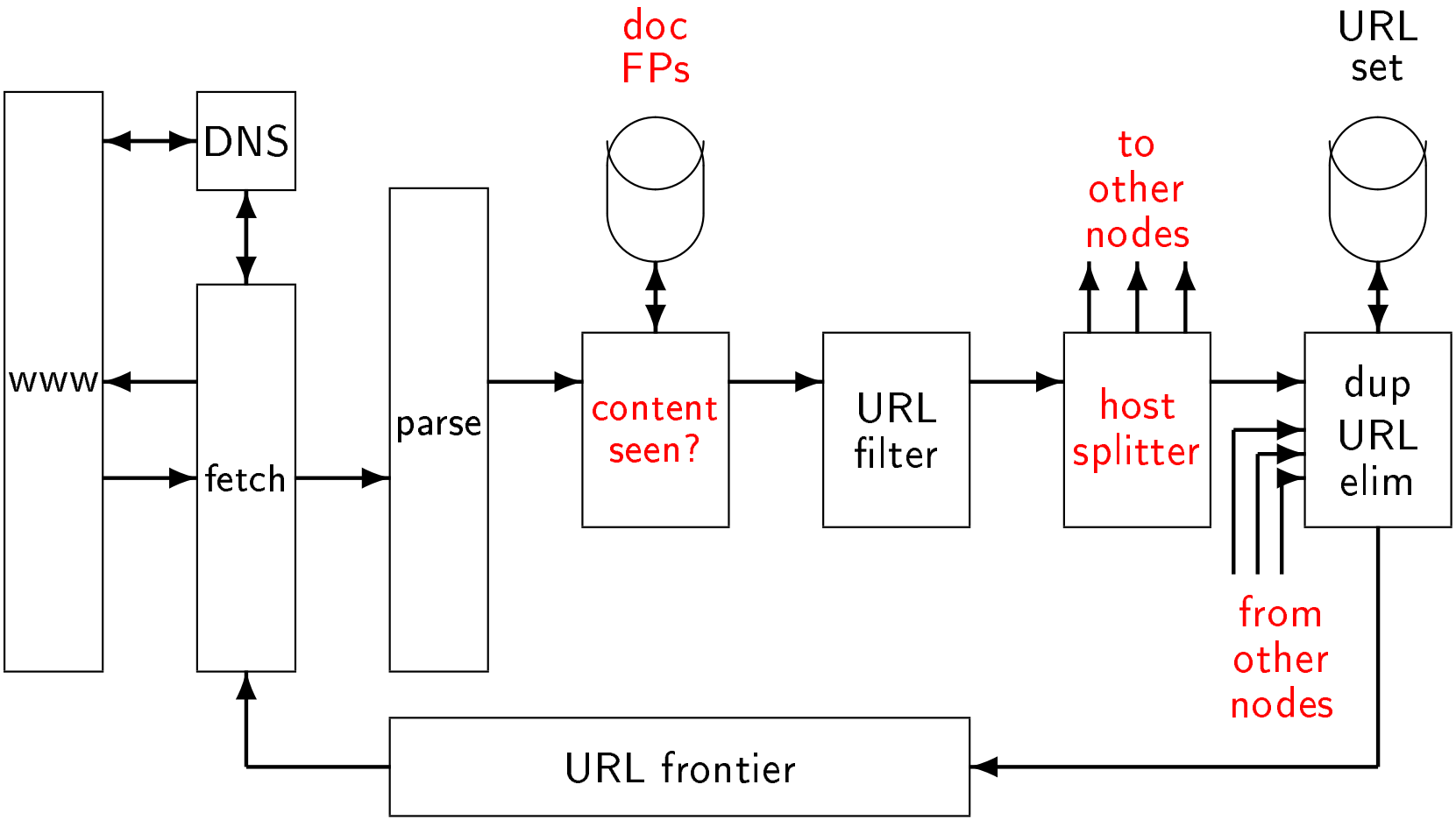


Mercator's main components.

Verteilter Crawler

- Um einen großen Anteil des Internets (in vertretbarer Zeit) zu spidern, ist eine Verteilung des Crawlers auf mehrere Maschinen unumgänglich
- Partitionierung über Hash auf Hostname
- Kommunikation zwischen Maschinen nötig, damit die zu crawlenden URLs verteilt werden

Distributed crawler [1]



Open Source Spider

- nutch
- grub (grub.org)
- Heritrix
- Apache Droids
- Aperture


H. S. Christopher Manning, P. Raghavan.
Introduction to Information Retrieval.
Cambridge, 2008.


A. Heydon and M. Najork.
Mercator: A scalable, extensible web crawler.
World Wide Web, 2(4):219–229, 1999.