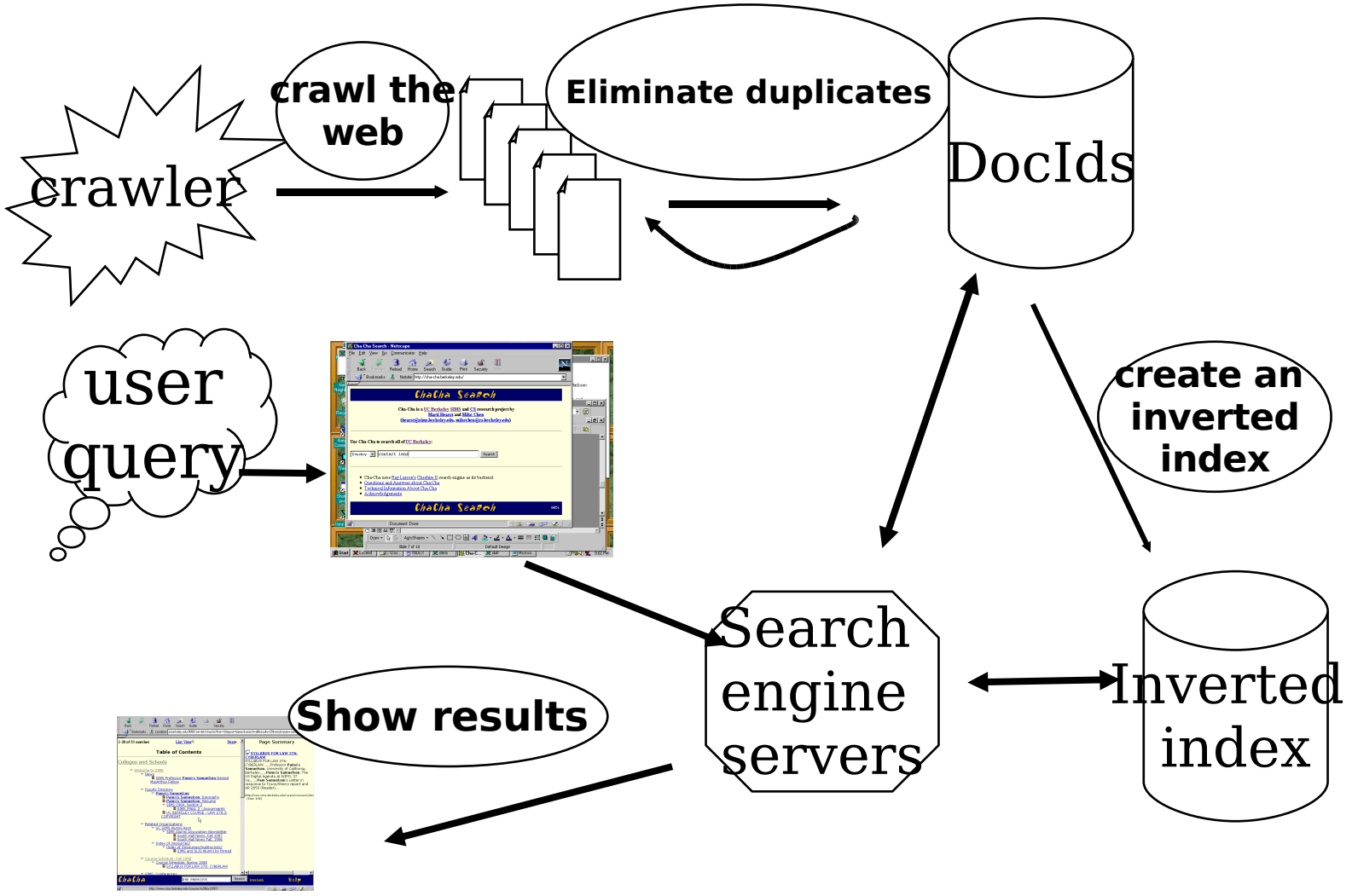
Florida International University

Mercator: A Scalable, extensible Web Crawler

by A. Heudon and M. Najork

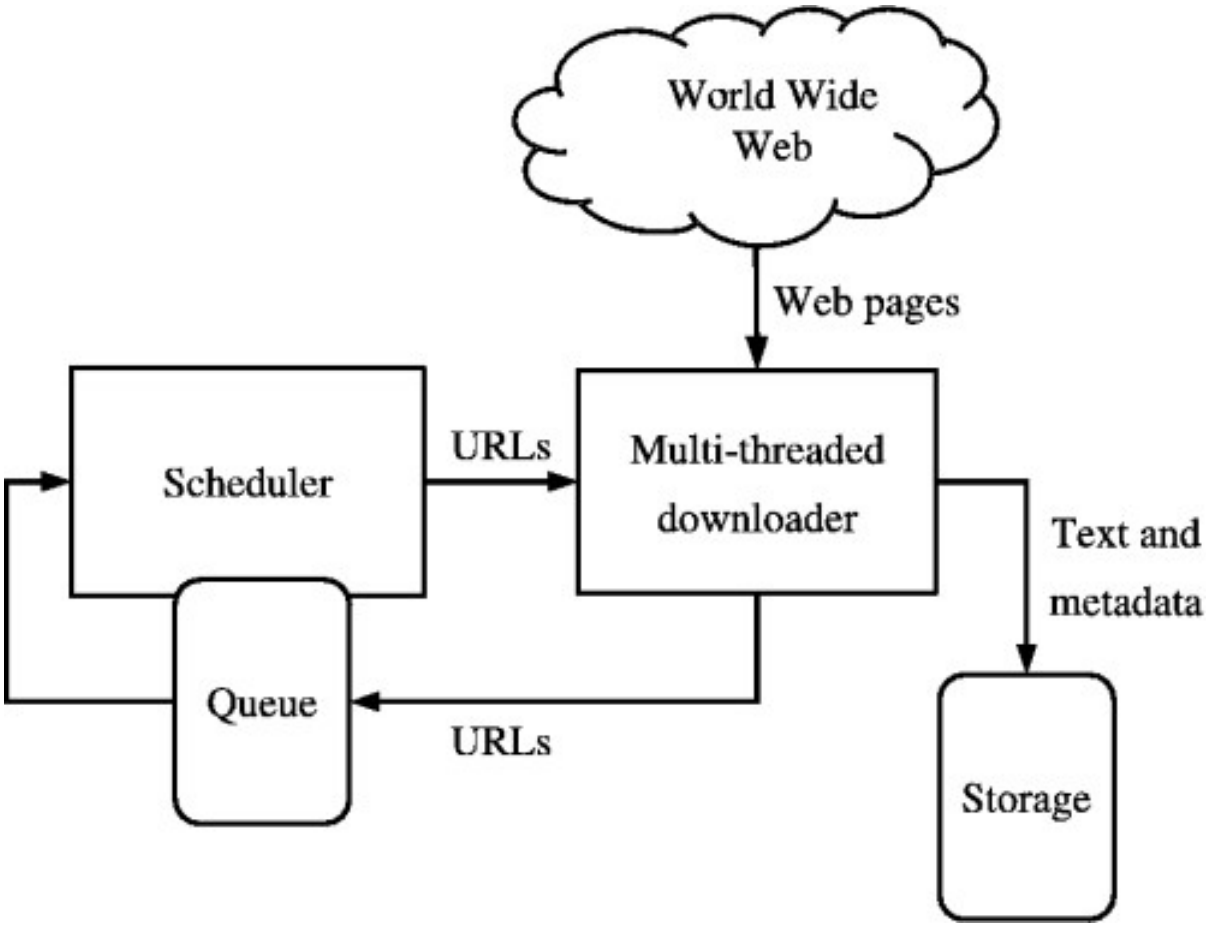
Presented by: L. Useche and R. Koller

Introduction



Web Crawler

- A web crawler is the entity in an IR system which browses the WWW.



Web Crawler Challenges

- **Performance**: How do you crawl 10,000,000,000 pages?
- **Politness**: How do you avoid overloading servers?
- **Legal**: What if the owner of a page does not want the crawler to index it?

Web Crawler Challenges (Cont)

- **Failures**: Broken links, time outs, spider traps.
- **Heterogeneous nature**: How do we manage different protocols or file types?

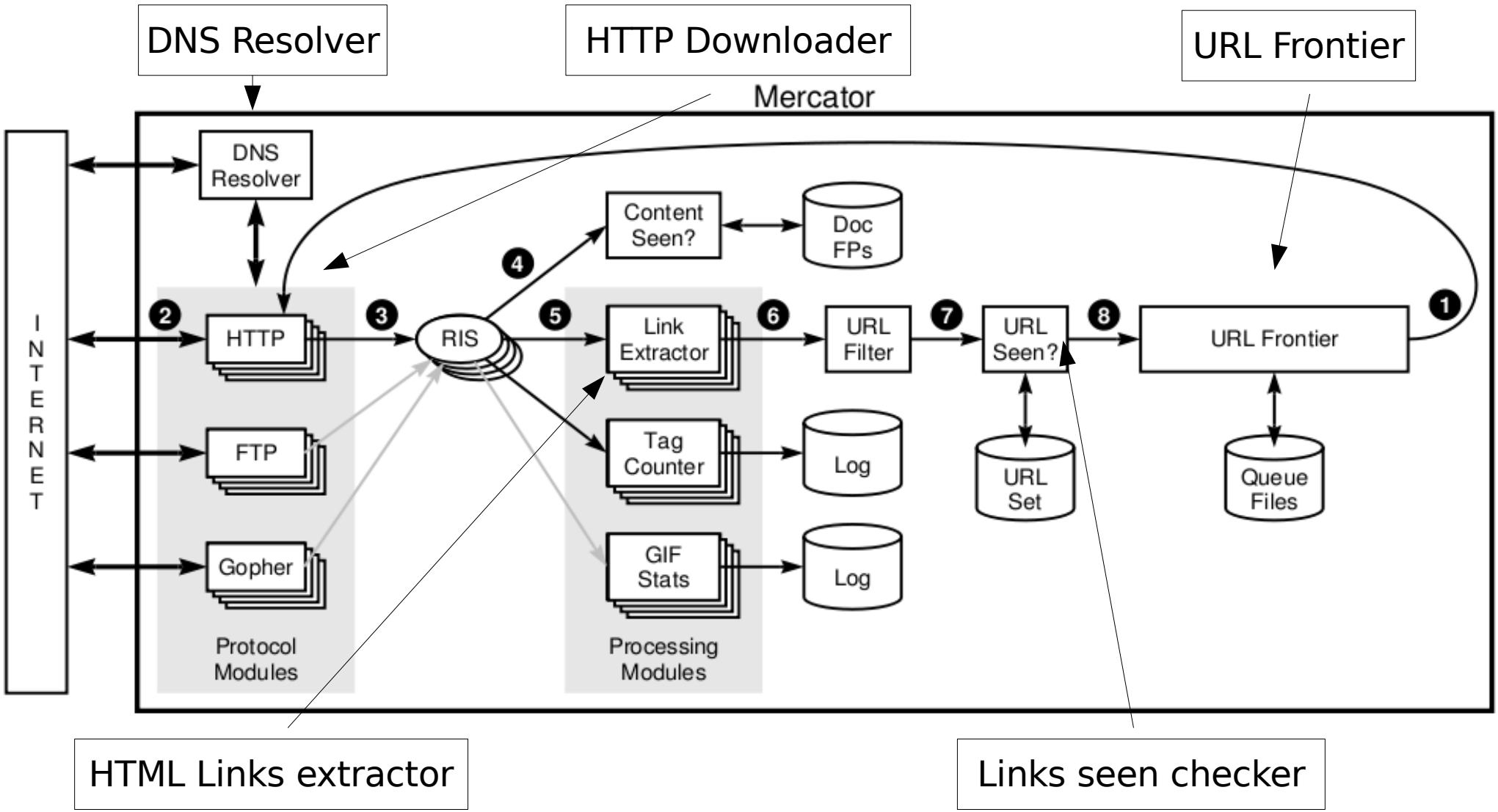
Mercator: Goals

- **Scalable**: capable to crawl tens of millions of web documents.
- **Extensible**: Modular architecture.
- **Distributed**: Capable to run in several machines.

Mercator: Goals (cont)

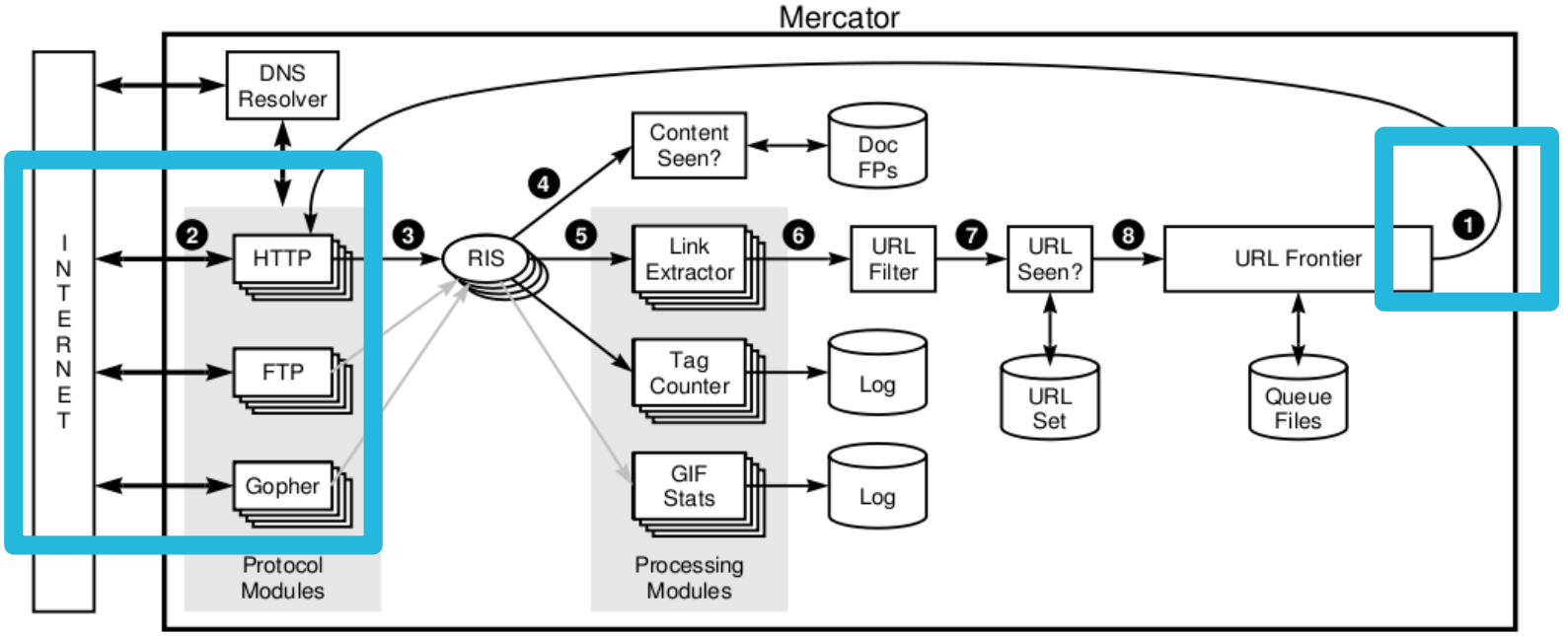
- **High performance**: the only limitation should be the internet connection.
- **Polite**: It is considered socially unacceptable to have multiple connections to the same server .

Mercator: Architecture Overview



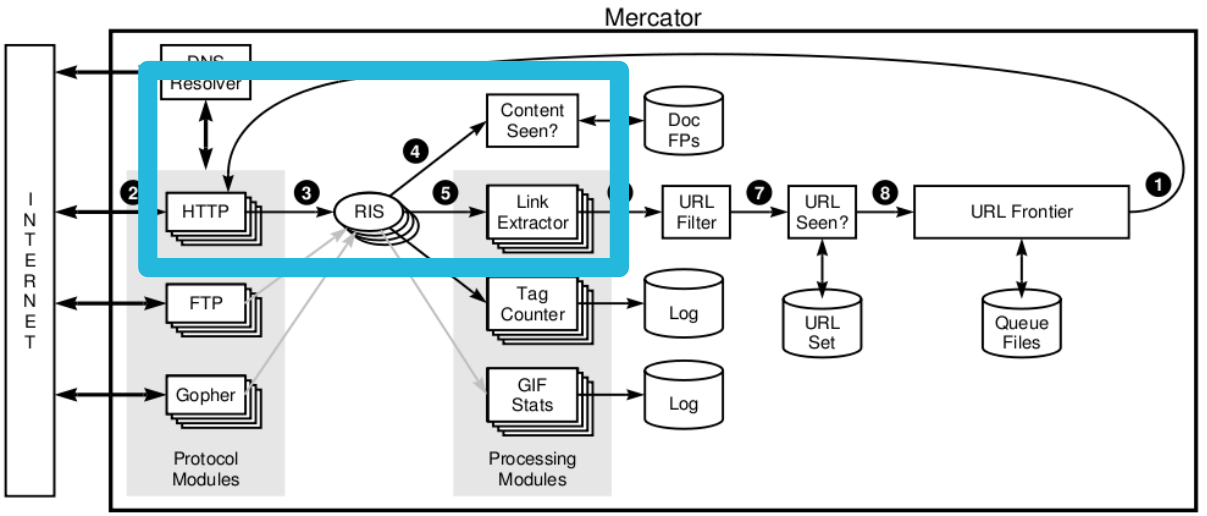
Mercator: Threads steps

- Obtain a URL from URL Frontier.
- After identifying the URL protocol, the worker downloads the file.



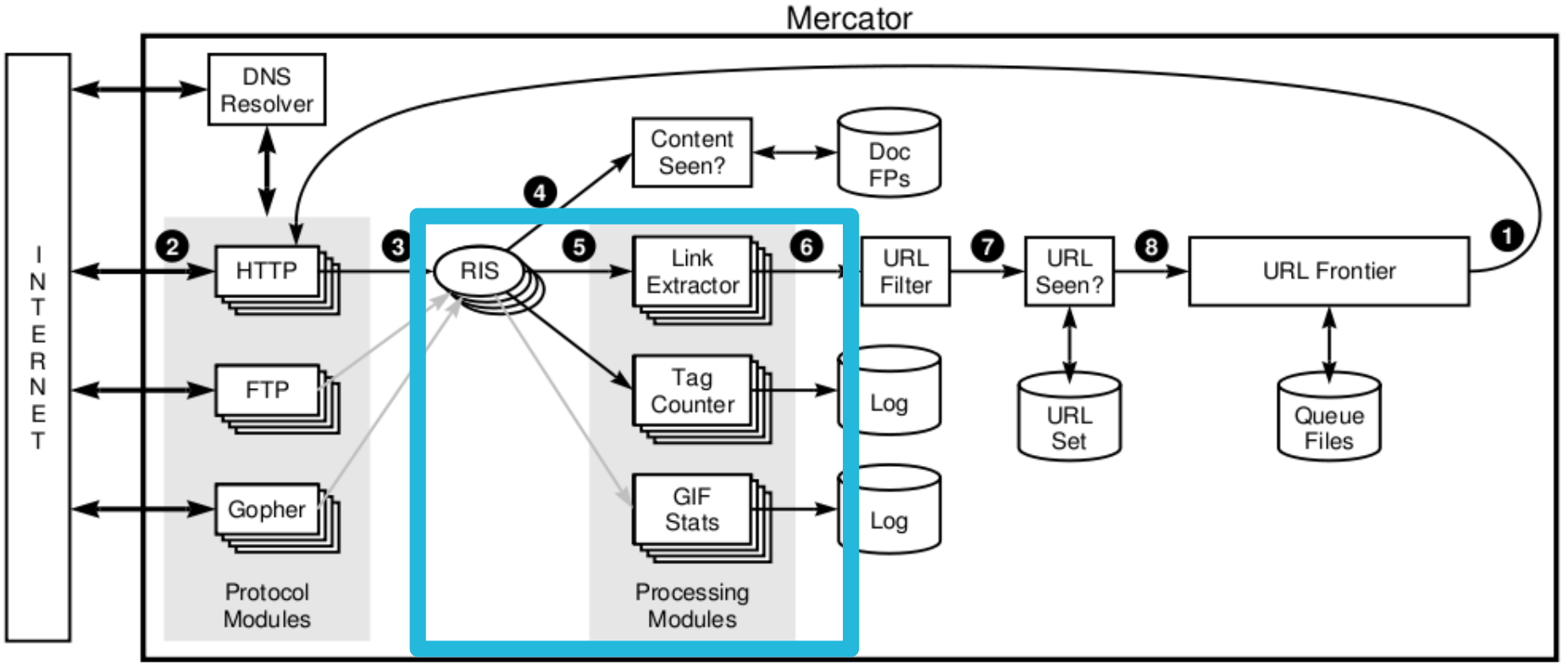
Mercator: Threads steps

- The file is placed in the Rewind Input Stream buffer.
- The *content-seen test* is invoked to determine if the document was already downloaded.



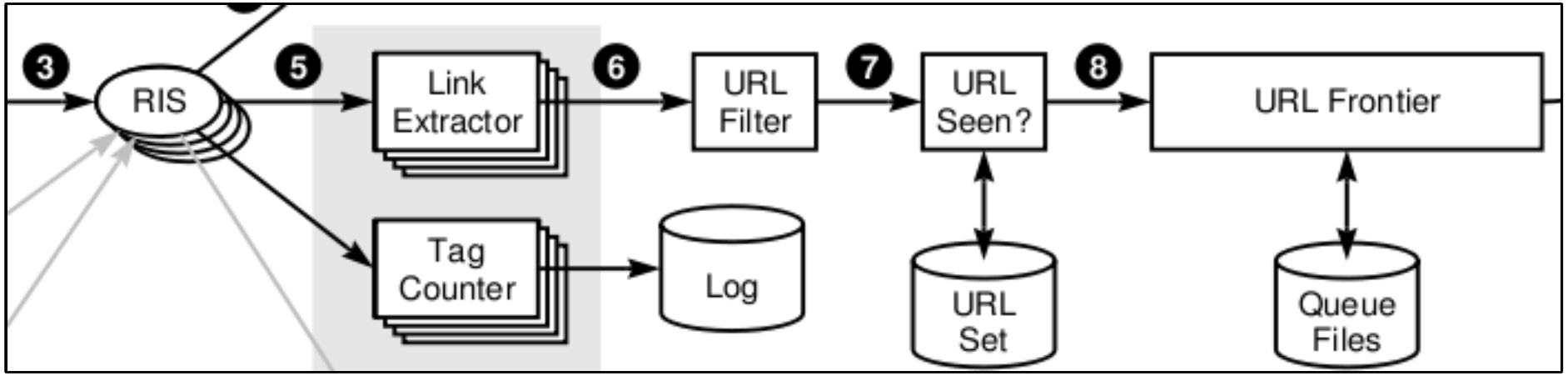
Mercator: Threads steps (cont)

- Based on the document type, the corresponding processor is called.



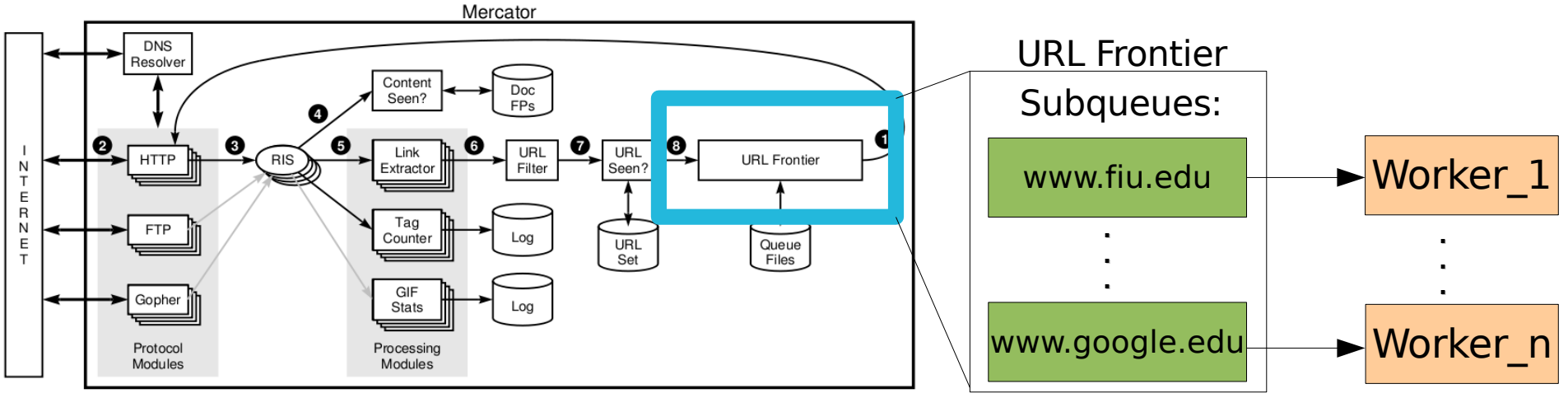
Mercator: Threads steps (cont)

- For each HTML document:
  - The links are extracted and converted to absolute URL.
  - The *URL-seen test* checks if the URL was already visited.
  - If not, the new URL is introduced to the frontier.



Mercator Components: URL Frontier, Politness

- Implemented as multiple queues.
- Each worker removes URLs from exactly one queue.
- The queue where the URL is added depends of its canonical host name.

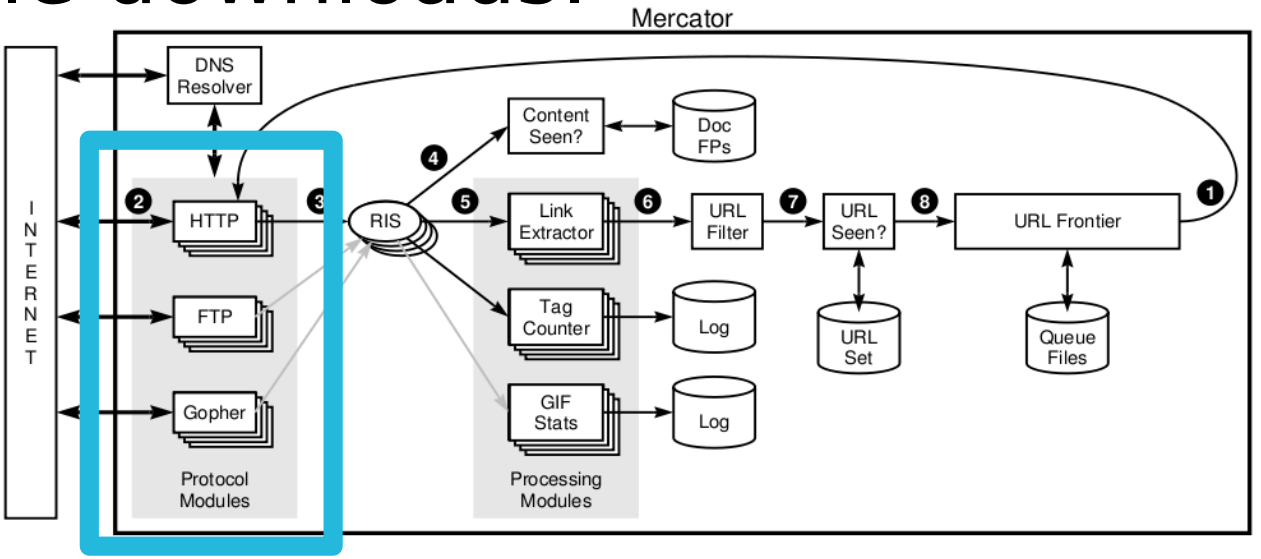


Mercator Components: URL Frontier (cont)

- In the current WWW crawls the size of the Frontier is in the order of hundreds of millions.
- Thus, Mercator has a enqueue and dequeue buffers of 600 URLs for each subqueue. The rest is in disk.

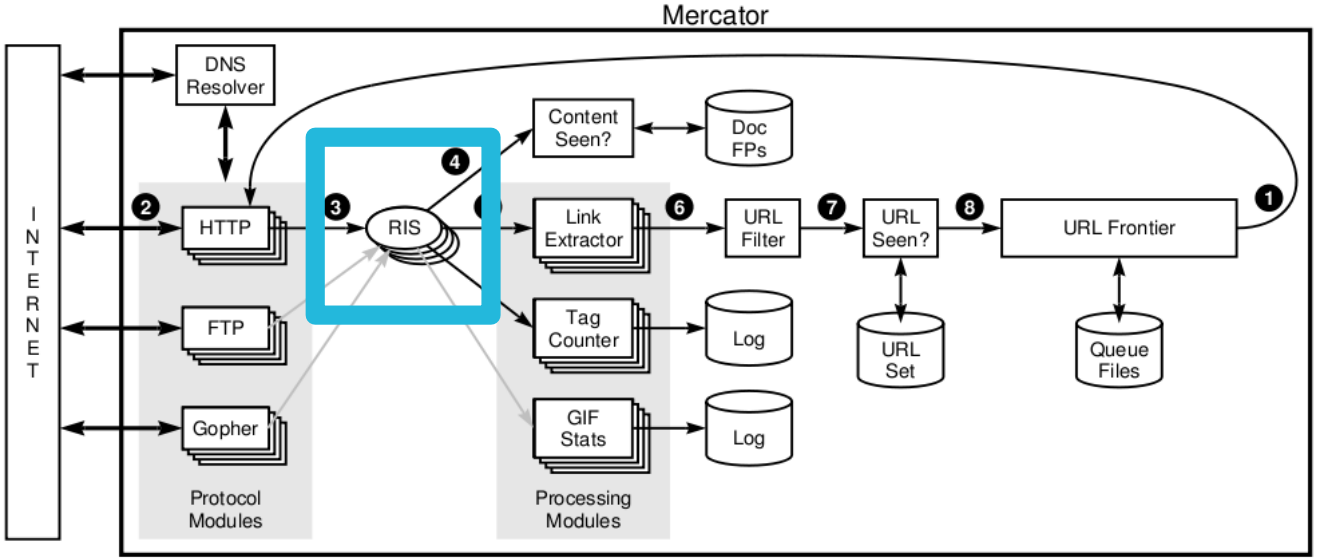
Mercator Components: Protocol Modules

- Mercator is capable to manage different protocols. Default: HTTP, FTP and Gopher.
- The HTTP module mantains a cache of each Robot Exlusion Rules to avoid multiple downloads.



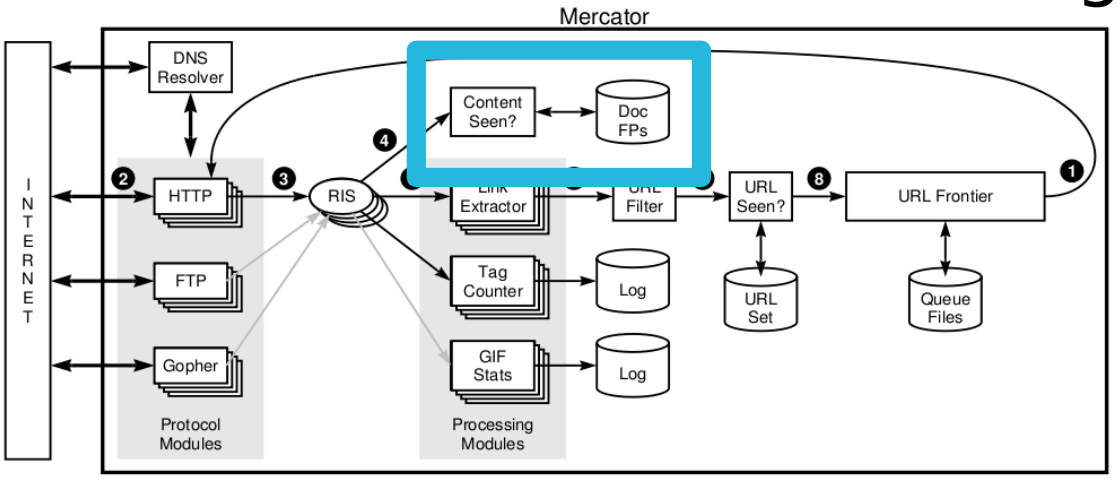
Mercator Components: Rewind Input Stream (RIS)

- Component that keep the data of the file buffered.
- RIS could cache in memory small files.
- Used by each module to process the file.



Mercator Components: Content-Seen Test

- There are multiple copies of one document in different URLs.
- Content-Seen Test keeps the fingerprint of all the documents.
- Each document is represented in a 64-bits checksum obtained with the Rabin's fingerprint algorithm.

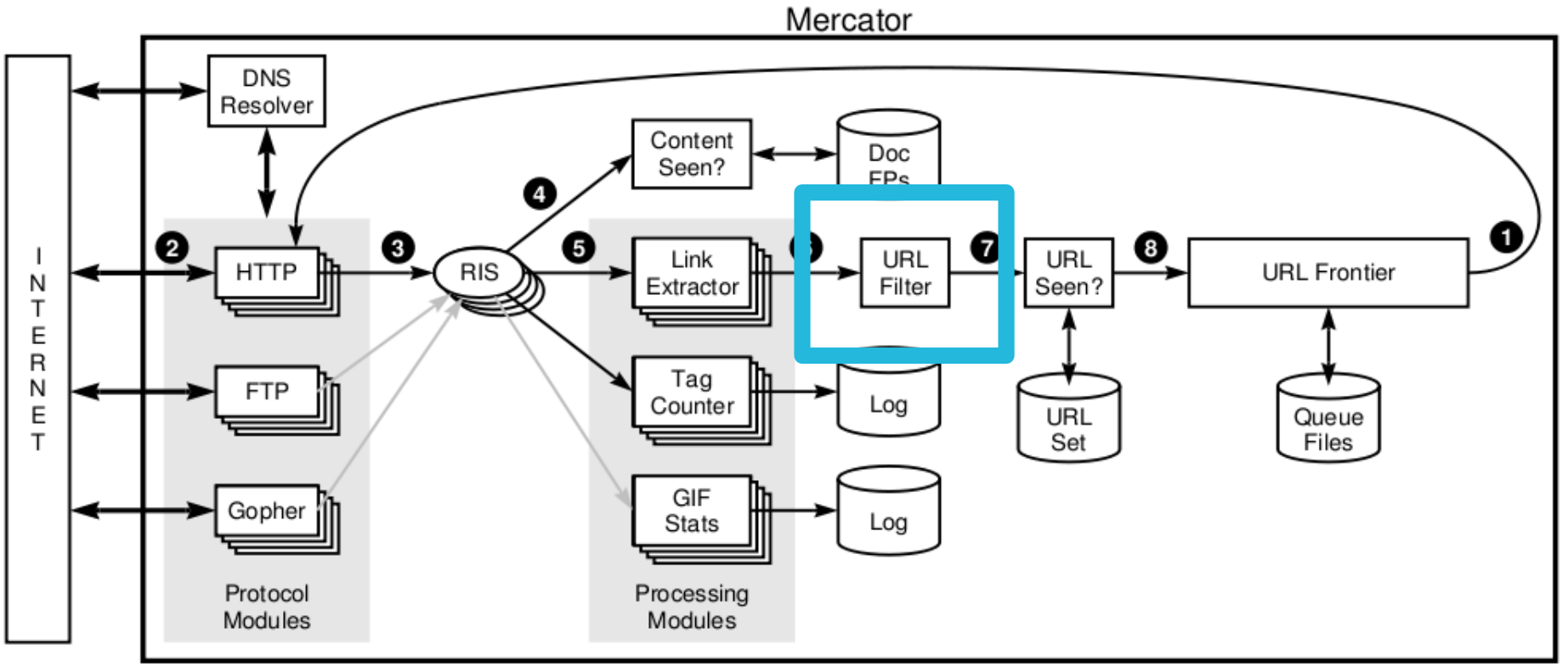


Mercator Components: Content-Seen Test (cont)

- All the fingerprints are stored in disk.
- Unfortunately, there is no locality for this component and the caching is useless.
- Mercator maintains an in-memory index for the finger prints file.

Mercator Components: URL Filters

- Mercator provides a URL filtering mecanism to skip some websites.

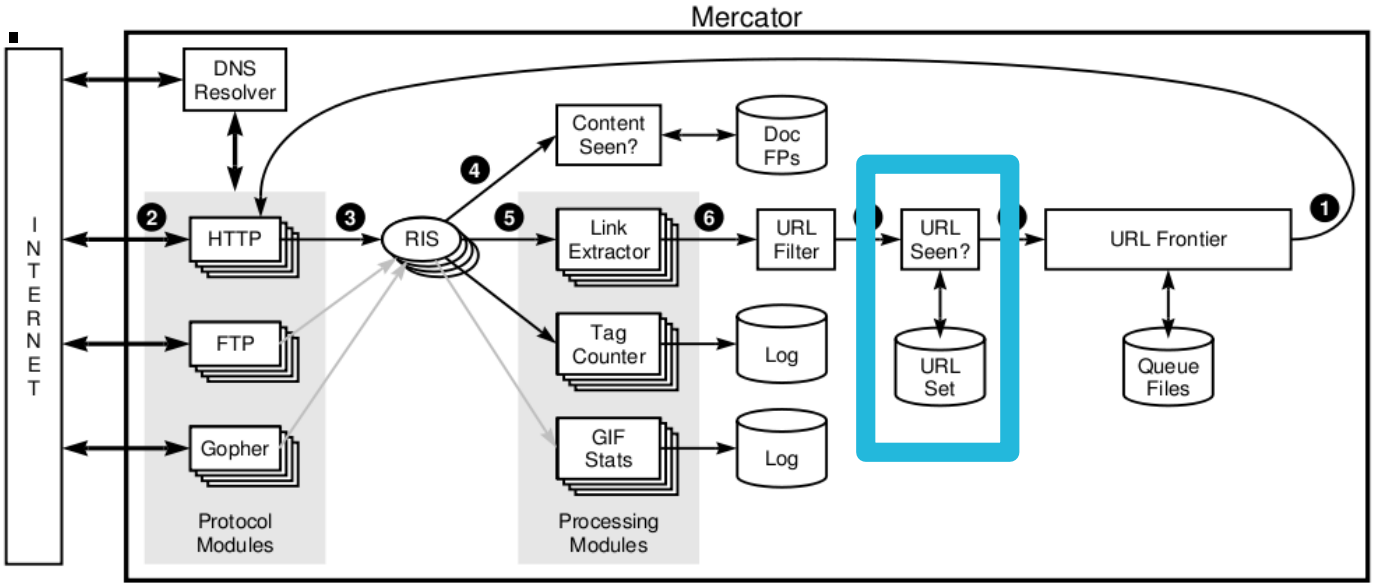


Mercator Components: IP resolution (DNS)

- DNS is a well known bottleneck in web crawlers.
- Mercator uses DNS cache.
- Java DNS lookups is synchronized.
- Solution: Mercator has its own implementation of DNS resolver.

Mercator Components: URL-Seen Test

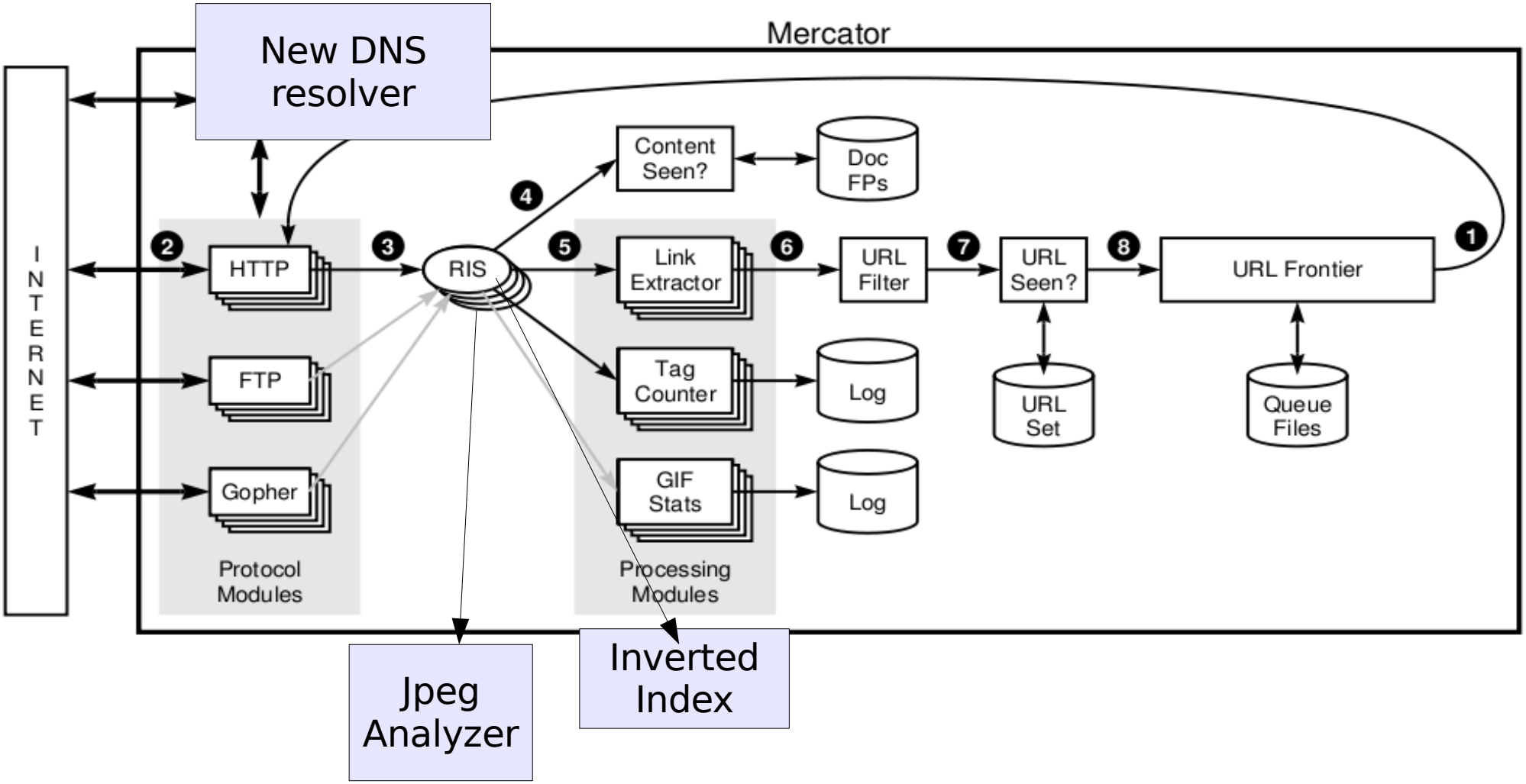
- The fingerprint of each URL is stored in disk like in Content-Seen Test.
- Fortunately, In this case there is a high locality and the use of a cache is very useful.



Checkpoint

- The crawl of the web could take weeks.
- **What if the crawler crash? we need to start the process again...**
- In order to avoid this problem, Mercator makes use of checkpoints.
- Thus, an interrupted or aborted crawl could be resume from the last checkpoint.
- The checkpoints is taken in daily basis.

Extensibility



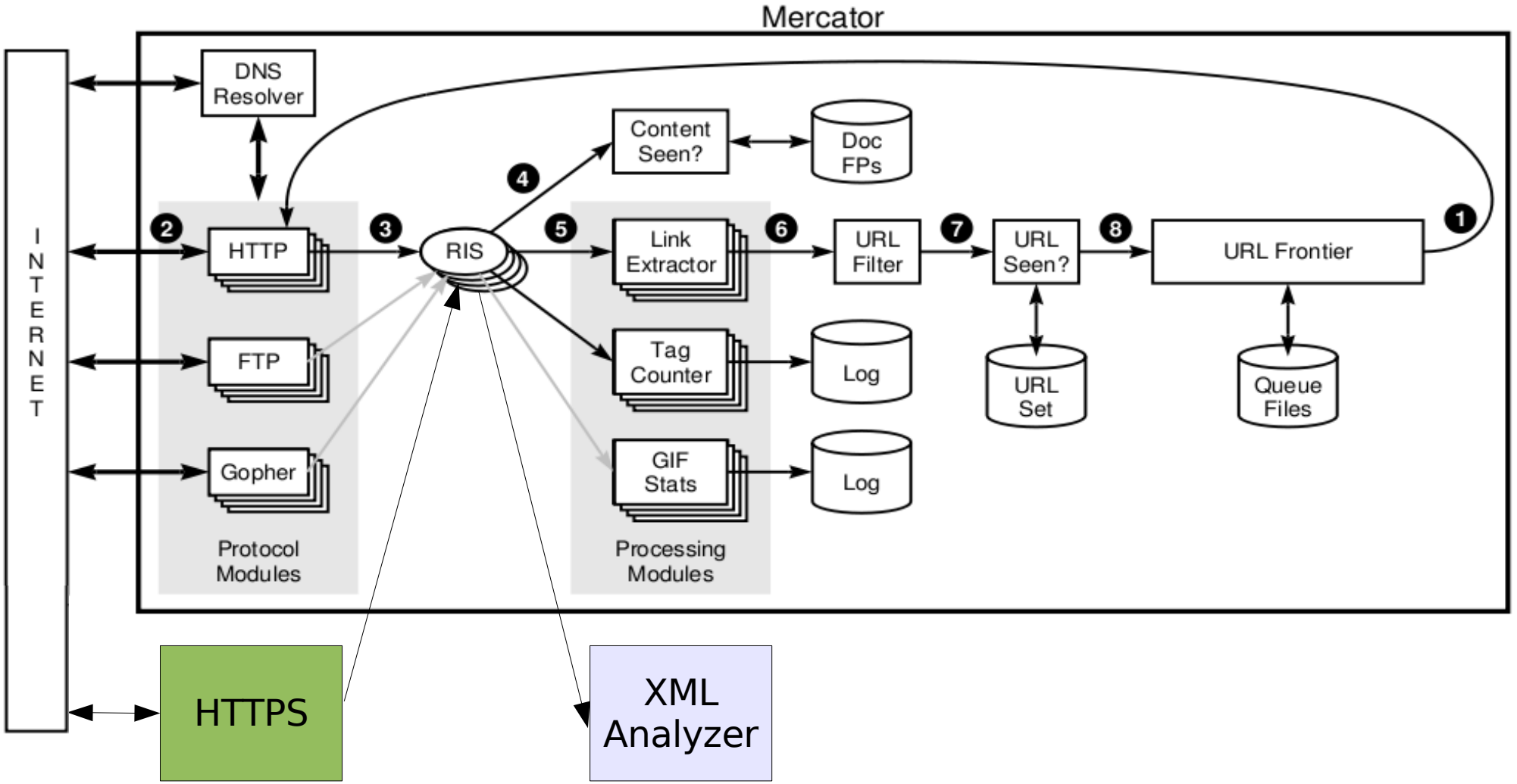
How does it achieve extensibility?

1)Well designed OO system

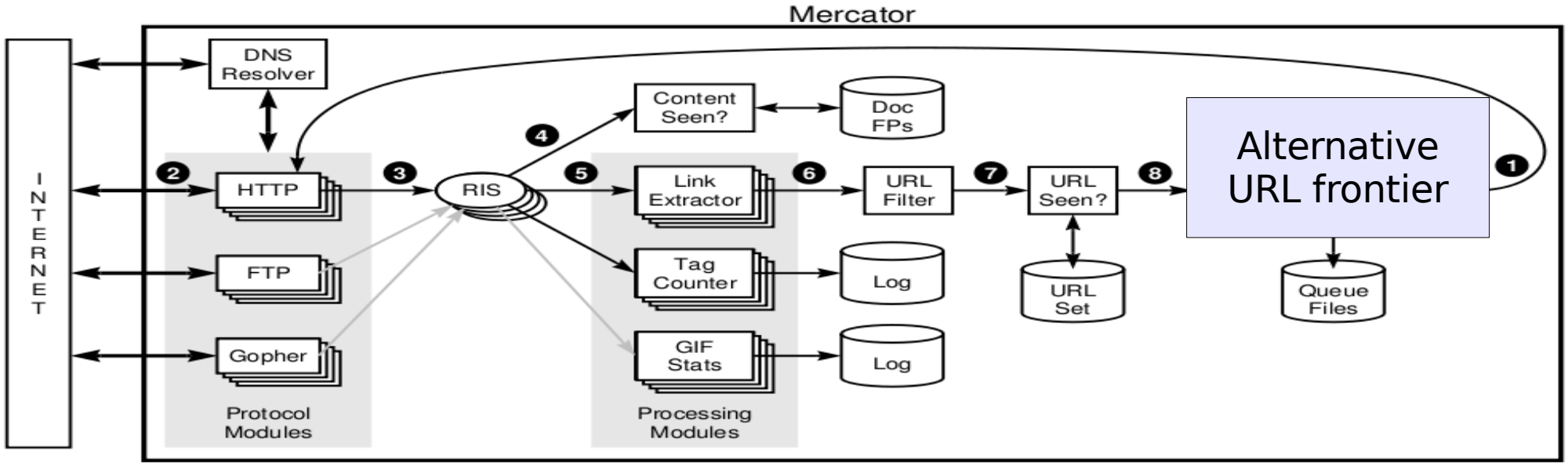
2)A configuration file

3)Infrastructure for new components

Ext. 1: New Protocols & Processing modules



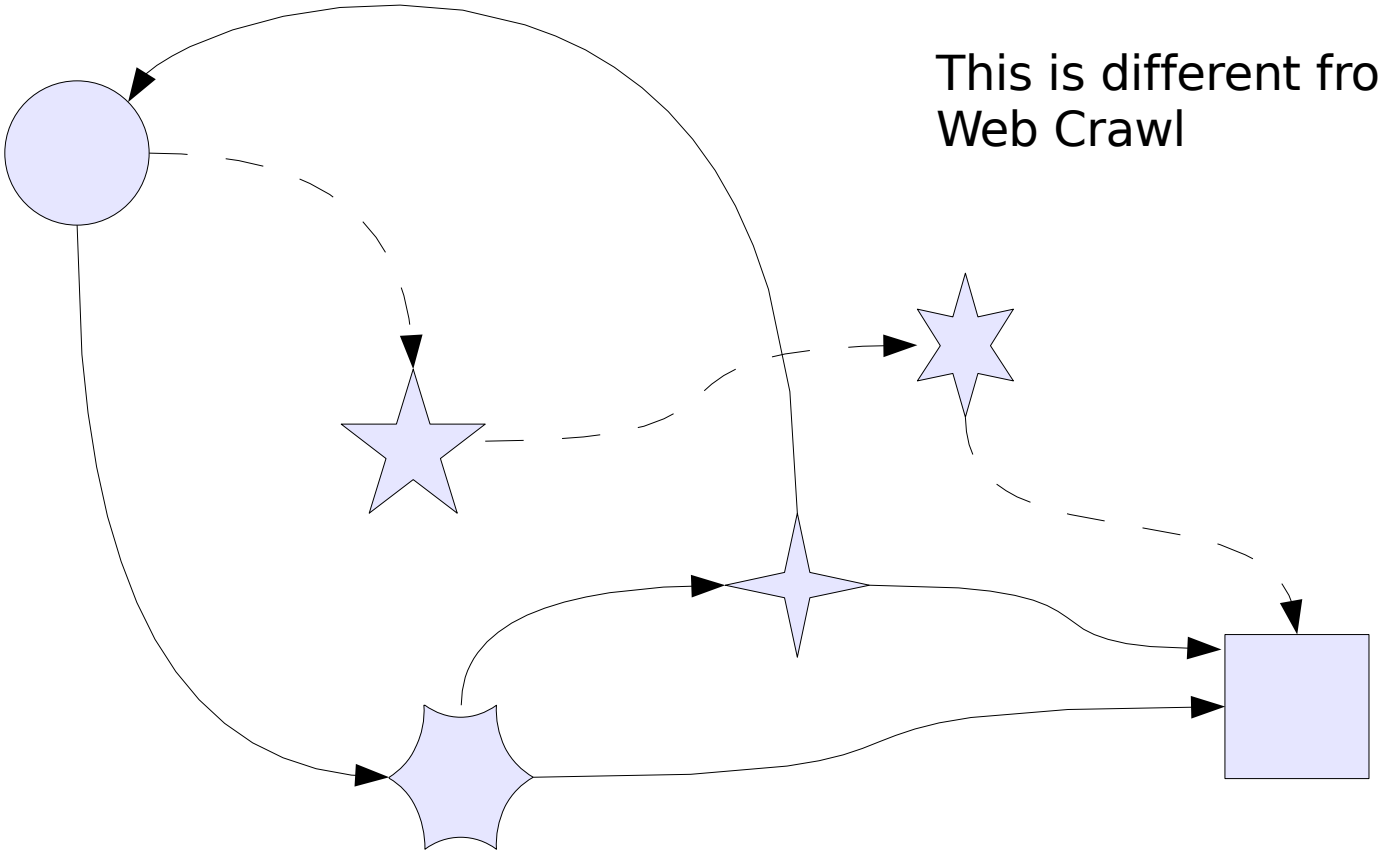
Ext. 2: URL frontier for Intranets



host1/a.html	host1/b.html	host3/b.html
host2/a.html		

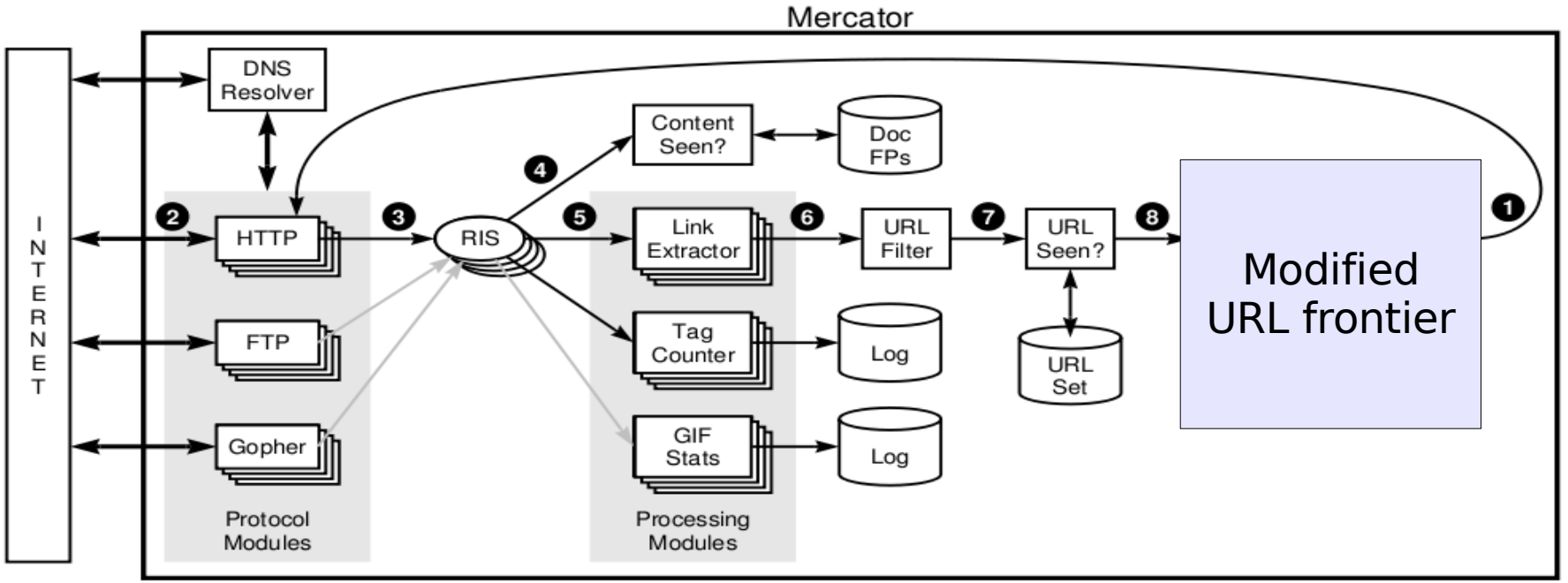
A new URL frontier

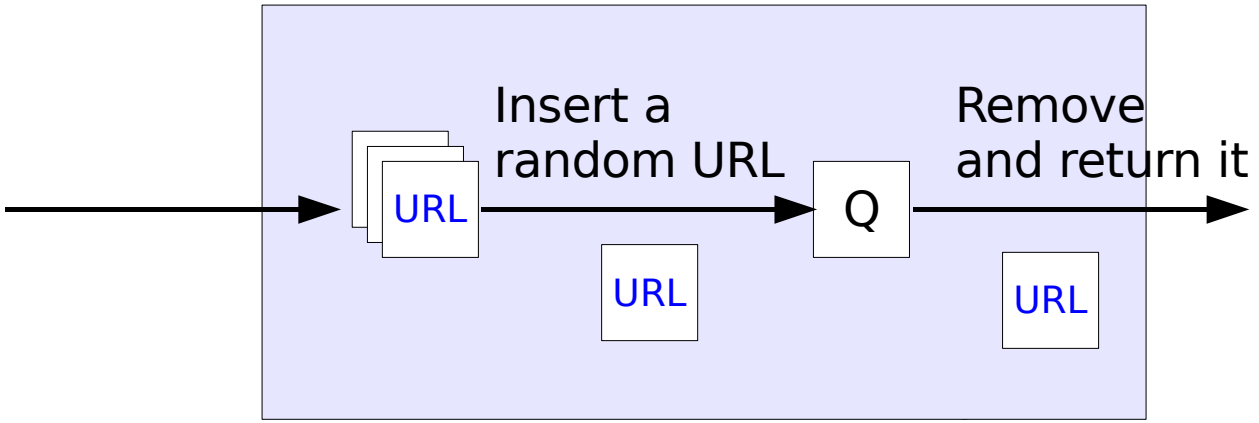
A Random Walk



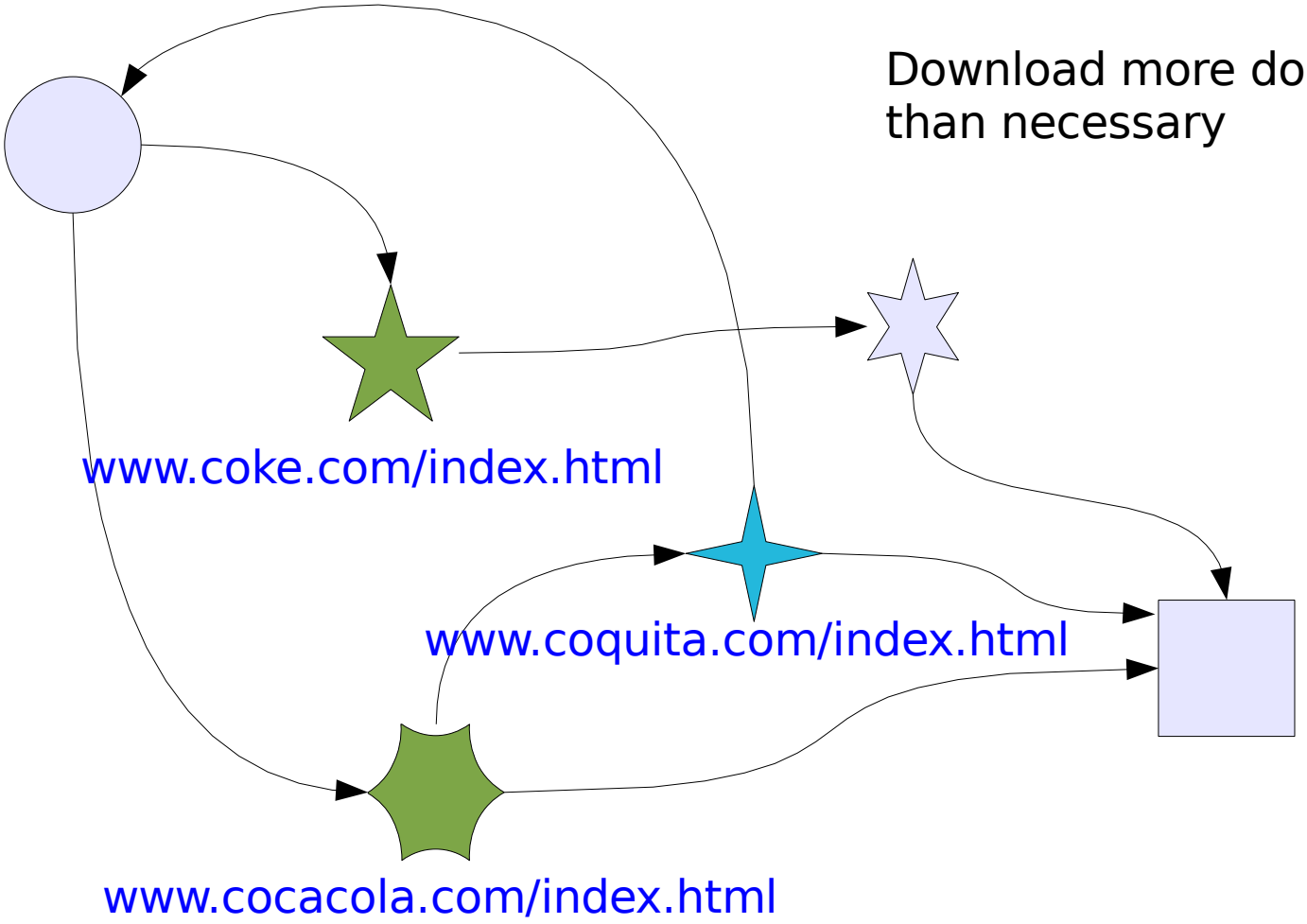
This is different from a Web Crawl

Ext. 3: Random Walker





Crawling hazard



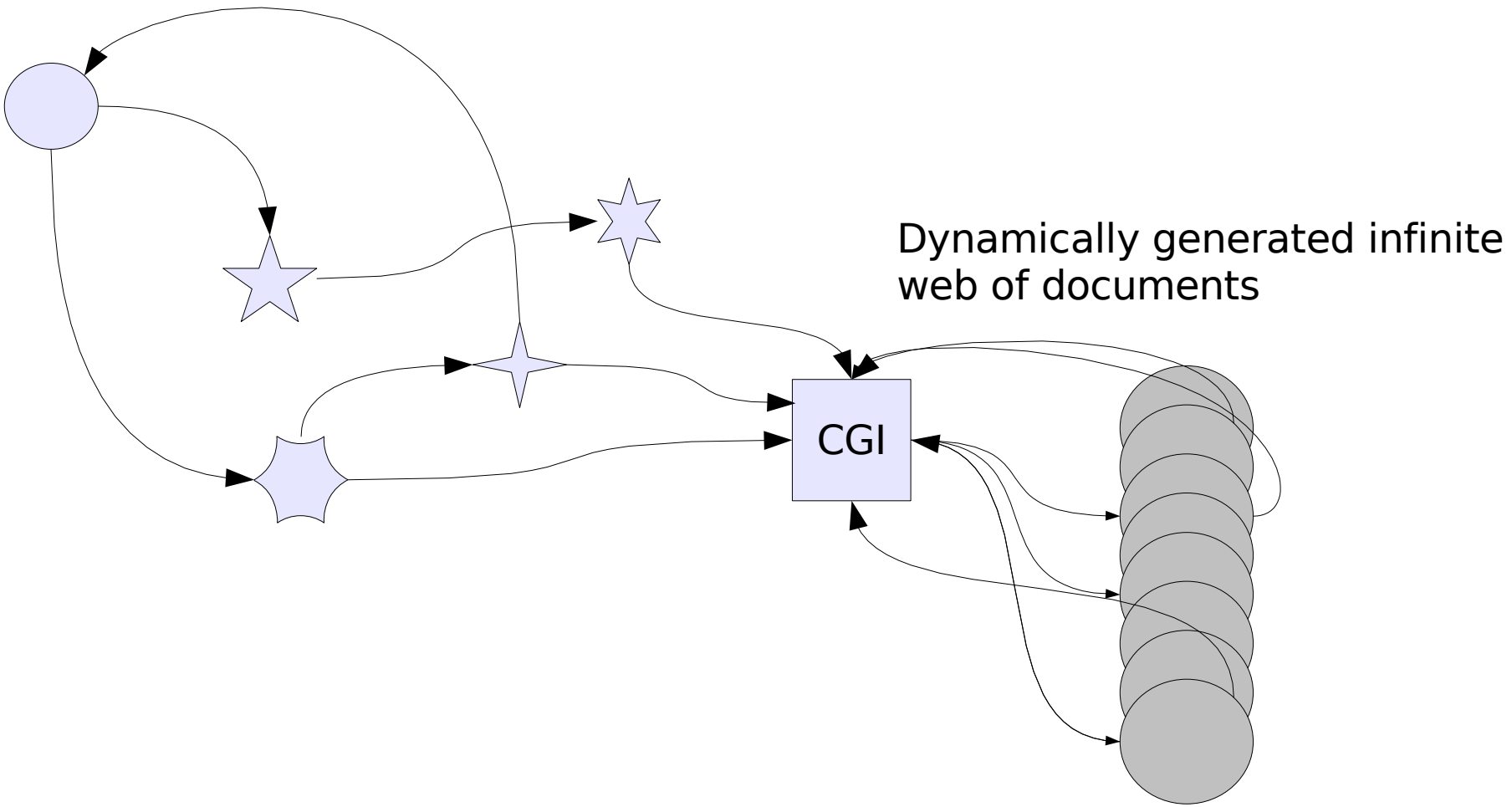
Prob. 1: URL Aliases

- Host name aliases
  - www.coke.com = www.cocacola.com
- Omitted port numbers
  - www.google.com:80 = www.google.com
- Alternative paths on the same host
  - 123.com/index.html = 123.com/home.htm
- Replication across different hosts
  - 1.com/index.html = 2.com/index.htm

Prob. 2: Session IDs embedded in URLs

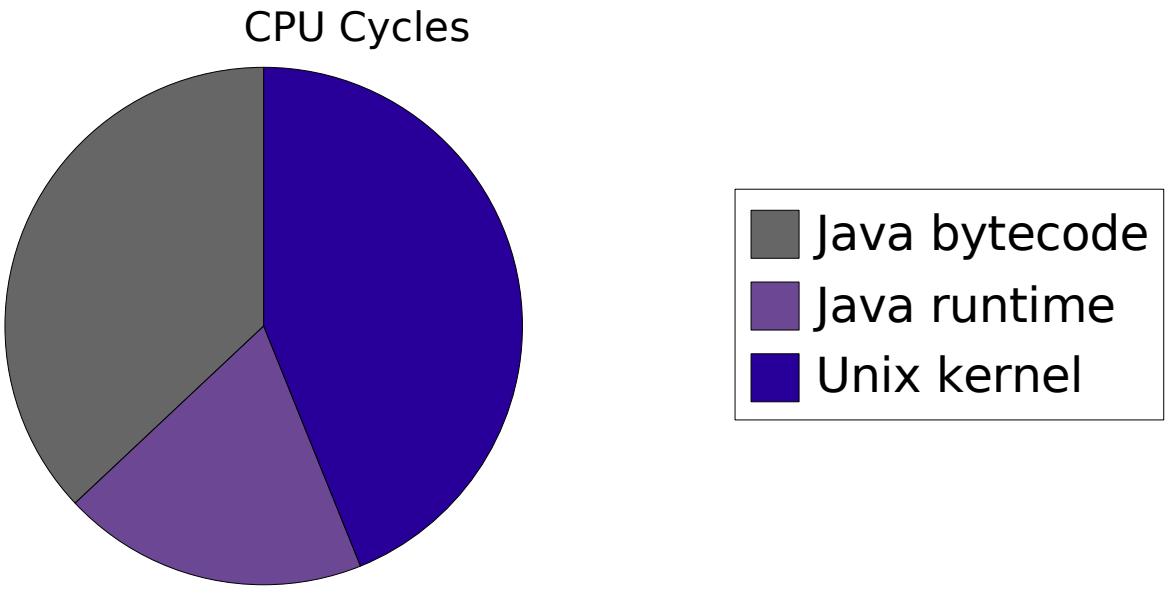
- Example:
  - website.com/a.asp?sessionid=bHbHNramR
- An infinite set of URLs refers to the same document
  - website.com/a.asp?sessionid=0000000000...
  - website.com/a.asp?sessionid=zzzzzzzzzzz..

Prob. 3: Crawler Traps



Results on Performance

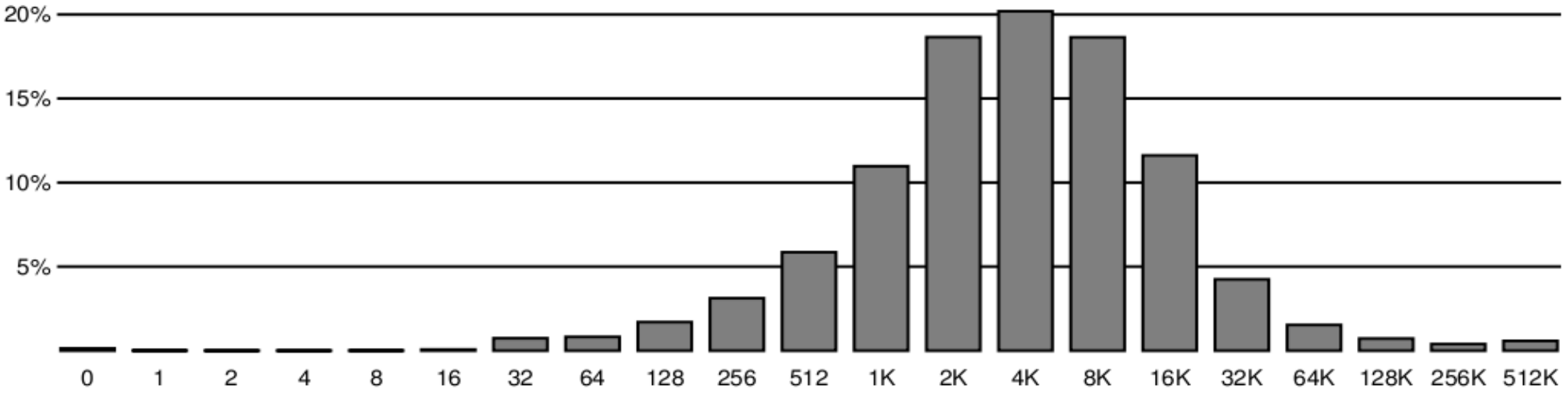
Crawler:	docs/sec	KB/sec
Google	33.5	200
Web Archive	46.3	231
Mercator	112	1682



Stat. 1: Breakdown of HTTP status codes

Code	Meaning	Number	Percent
200	OK	65,790,953	87.03%
404	Not Found	5,617,491	7.43%
302	Moved Temporarily	2,517,705	3.33%
301	Moved Permanently	842,875	1.12%
403	Forbidden	322,042	0.43%
401	Unauthorized	223,843	0.30%
500	Internal Server Error	83,744	0.11%
406	Not Acceptable	81,091	0.11%
400	Bad Request	65,159	0.09%
	Other	48,628	0.06%
	Total	75,593,531	100.0%

Stat. 2: Size of documents



Stat. 3: Distribution of MIME types

MIME type	Number	Percent
text/html	41,490,044	69.2%
image/gif	10,729,326	17.9%
image/jpeg	4,846,257	8.1%
text/plain	869,911	1.5%
application/pdf	540,656	0.9%
audio/x-pn-realaudio	269,384	0.4%
application/zip	213,089	0.4%
application/postscript	159,869	0.3%
other	829,410	1.4%
Total	59,947,946	100.0%

Conclusions

- Building a scalable crawler is a non-trivial task
- Java as the implementation language was a good decision
- Extensibility features were successful
- Mercator was a success: it is now the crawler used by *Altavista Enterprise Search*