

# Crawler.NET: A component-based distributed framework for web traversal

Levente Hunyadi (BME AAIT)

March 23, 2007

Introduction

Motivation

Objectives

Architecture

Component  
framework

Crawler  
application

Conclusions

## The Web:

- a source of distributed information
- a giant set of semi-structured data

⇒ search engines are invaluable to locate information

Introduction

Motivation

Objectives

Architecture

Component  
framework

Crawler  
application

Conclusions

- up-to-date index database  
    ⇓
- efficient traversal  
    ⇓
- parallelization  
    ⇓
- distributed architecture  
    ⇓
- increased complexity

# Objectives

Introduction

Motivation

Objectives

Architecture

Component  
framework

Crawler  
application

Conclusions

- scalability
- easy configuration and management
- support for extension
- robustness, resilience to failures

# Architectural overview

Introduction

Motivation

Objectives

Architecture

Component  
framework

Crawler  
application

Conclusions

Two separate layers:

## Component framework

### General tasks

- component interaction
- lifecycle management
- transparent interprocess communication

## Crawling application

### Field-specific issues

- downloading documents
- extracting hyperlinks
- administering page references
- scheduling requests

Introduction

Motivation

Objectives

Architecture

Component  
framework

Crawler  
application

Conclusions

- the component framework exposes general component skeletons that realize common behavior
- new, field-specific components are created by means of inheritance
- the framework provides loose coupling between components

Advantages:

- + simpler and faster development
- + openness for extension

# Building blocks of the architecture

Introduction

Component  
framework

Building blocks

Components

Providers

Connectors

Crawler  
application

Conclusions

- **Components**  
encapsulate field-specific functionality, produce, consume or transform data
- **Providers**  
give access to data sources
- **Connectors**  
provide asynchronous, message-based communication between components

# Components

Introduction

Component  
framework

Building blocks

Components

Providers

Connectors

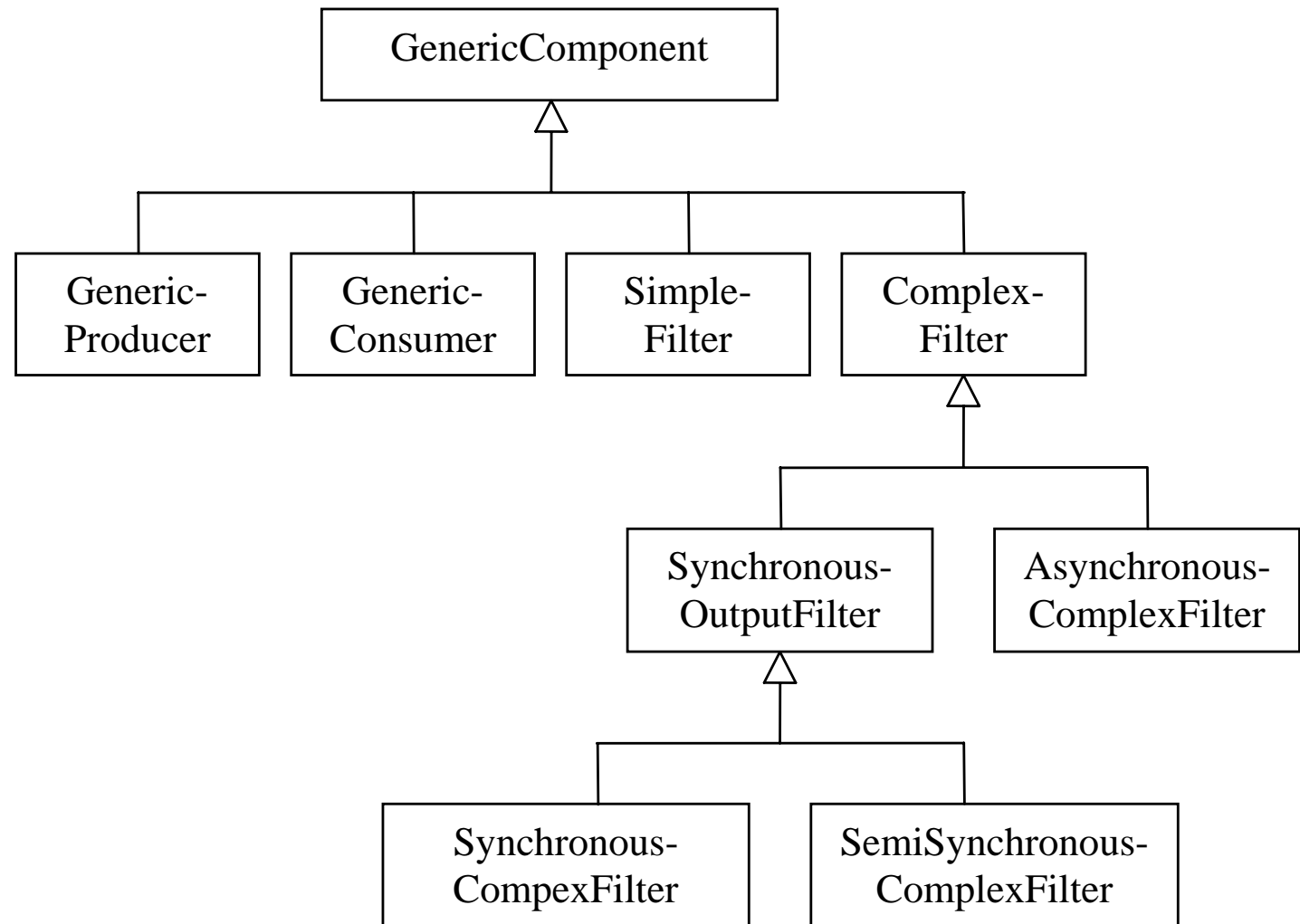
Crawler  
application

Conclusions

- abstract base class implements generic tasks
- differentiated subclasses based on how they interact with environment



# Components



Introduction

Component  
framework

Building blocks

Components

Providers

Connectors

Crawler  
application

Conclusions

- wrap external resources used by components
- synchronized access to data sources
- diverse functionality:
  - ☐ access databases
  - ☐ transparent cache mechanisms
  - ☐ network resources

Introduction

Component  
framework

Building blocks

Components

Providers

Connectors

Crawler  
application

Conclusions

- abstractions of typed queues
- represent a message queue
- intra-process or inter-process
- support one-to-many, many-to-many relationships, identification by roles

# Relization of connectors

Introduction

Component  
framework

Building blocks

Components

Providers

Connectors

Crawler  
application

Conclusions

Method of message transfer transparent to components:

- **local connector**

typed FIFO queue

data is passed *by reference*

- **remote connector**

corresponds to two local queues and associated network communication components in separate processes

data is *serialized* (and transmitted over TCP)

Introduction

Component  
framework

Crawler  
application

Architecture

Server  
components

Client  
components

Traversal

Load balancing

Parsing

URL distributor  
component

Conclusions

## Client-server architecture:

- **clients** retrieve documents with respect to the appropriate traversal strategy
- the **server** partitions the web and assigns partitions to clients

Implementation using component framework classes

# Marshaler component

Introduction

Component  
framework

Crawler  
application

Architecture

Server  
components

Client  
components

Traversal

Load balancing

Parsing

URL distributor  
component

Conclusions

- forwards incoming URLs to clients based on domain or host name
- caches recently forwarded URLs to decrease network load

# Marshaler component

Introduction

Component  
framework

Crawler  
application

Architecture

Server  
components

Client  
components

Traversal

Load balancing

Parsing

URL distributor  
component

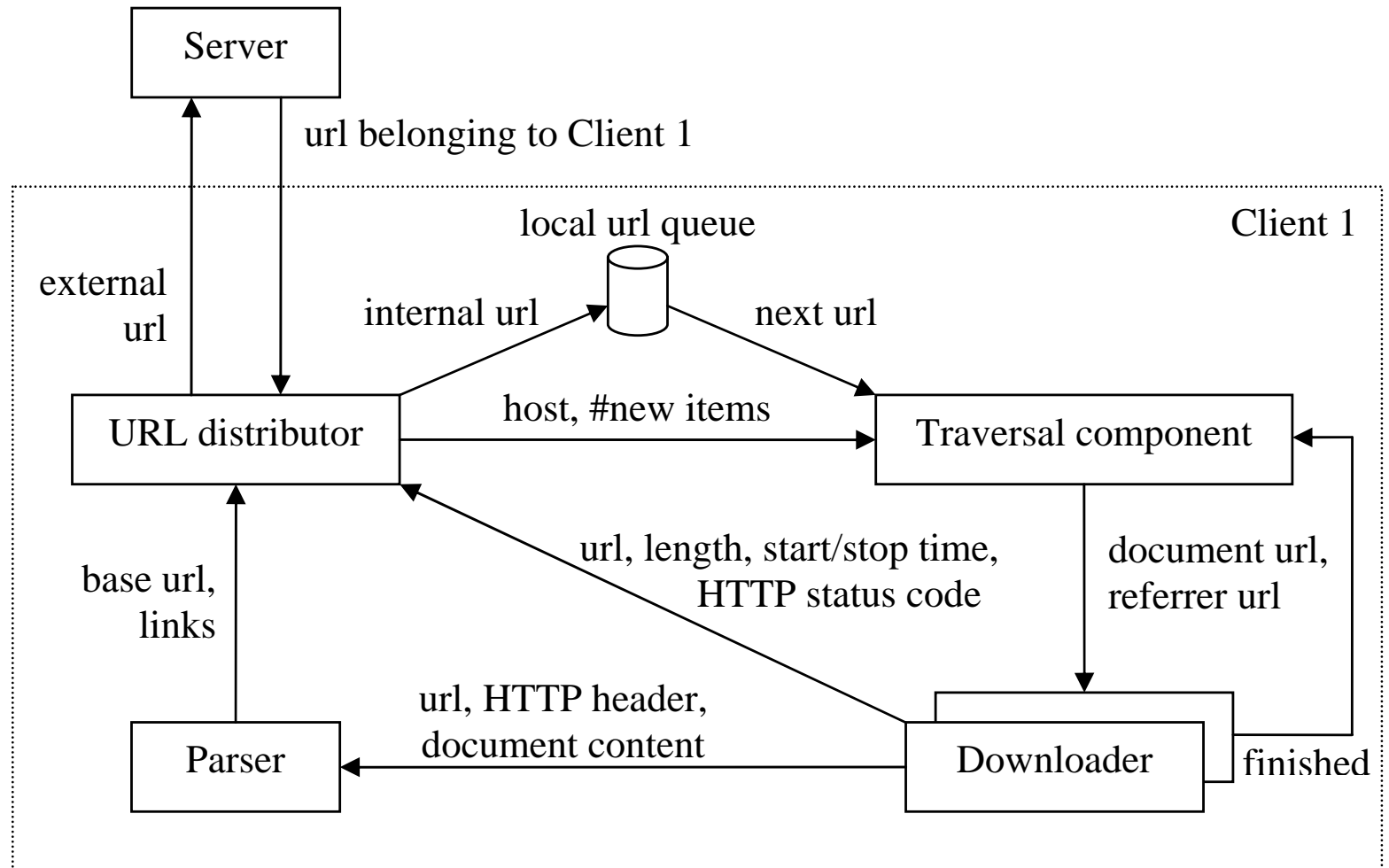
Conclusions

Limited data exchange during web traversal:

- *locality principle*: approx. 10% of hyperlinks are outbound from host or domain
- *batch transmission*
- *Zipfian distribution*: discarding cached URLs leads to sharply reduced load

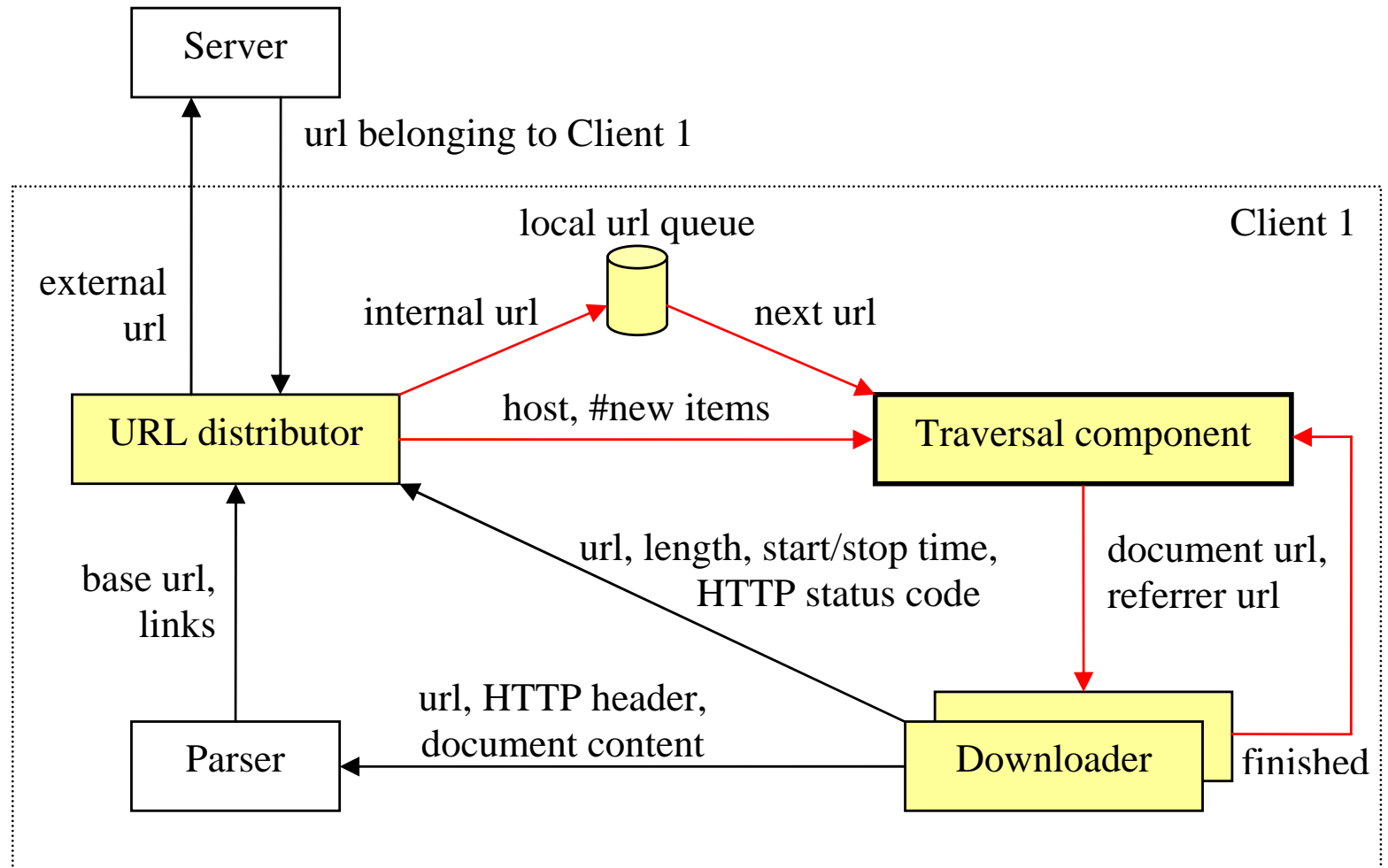
Load balancing between marshalers: URL distribution based on URL host name hash

# Basic client components





# Traversal component



# Traversal component

Introduction

Component  
framework

Crawler  
application

Architecture

Server  
components

Client  
components

Traversal

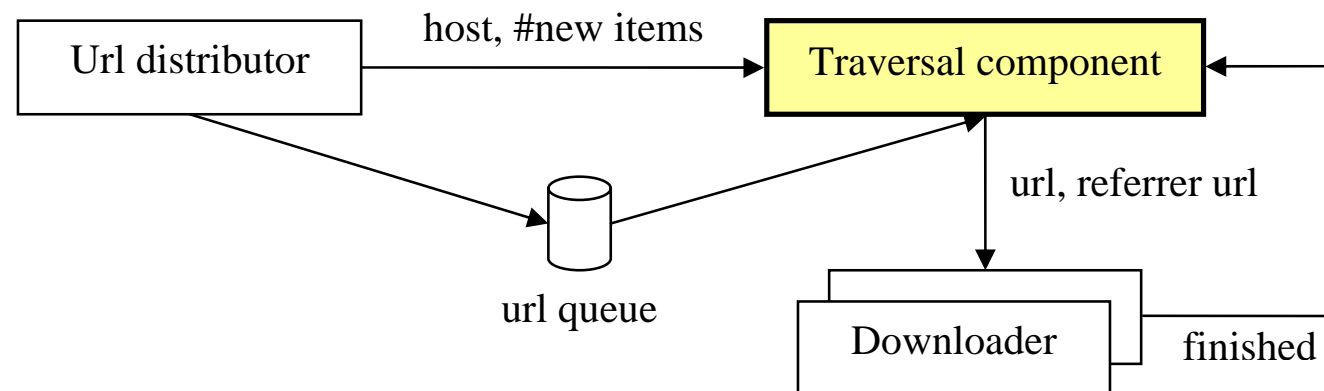
Load balancing

Parsing

URL distributor  
component

Conclusions

- fetches new URLs to download from *persistent storage*
- *notification* on arrival of new URLs from server or availability of a host
- selects next URL based on traversal strategy (breadth-first, relevance-based, etc.)



# Load balancing component

Introduction

Component  
framework

Crawler  
application

Architecture

Server  
components

Client  
components

Traversal

Load balancing

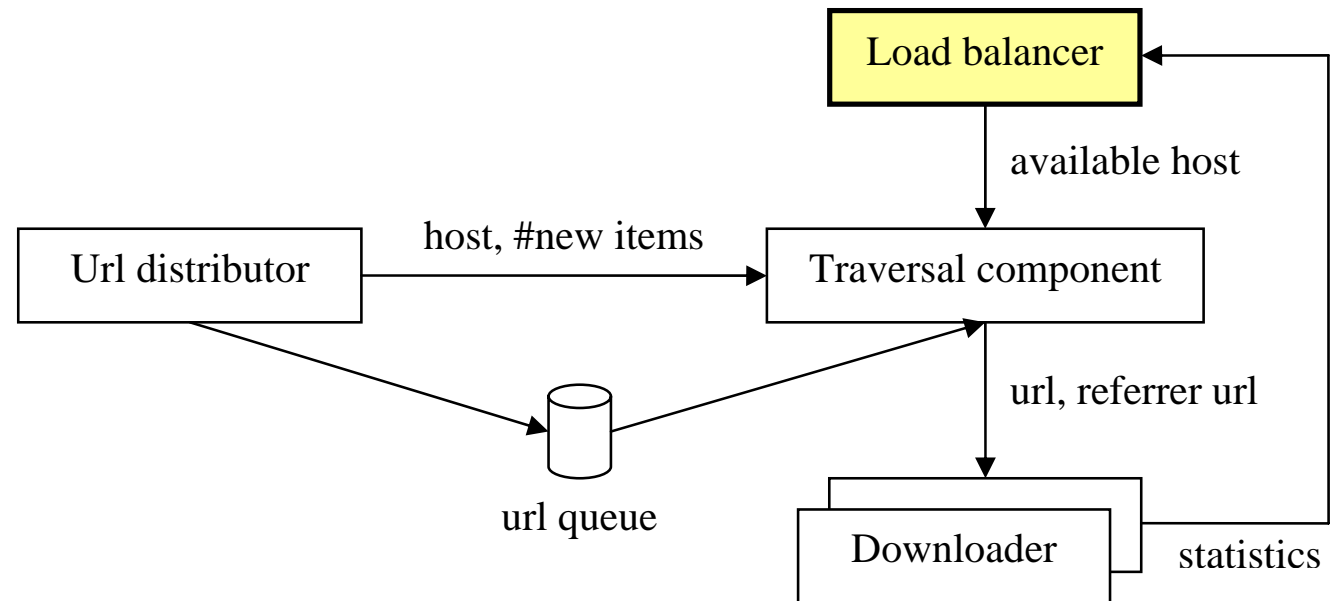
Parsing

URL distributor  
component

Conclusions

- prevents overloading hosts
- cooperates with traversal components
- configurable delay between requests
- dynamic adaptation based on response times

# Load balancing component



Introduction

Component  
framework

Crawler  
application

Architecture

Server  
components

Client  
components

Traversal

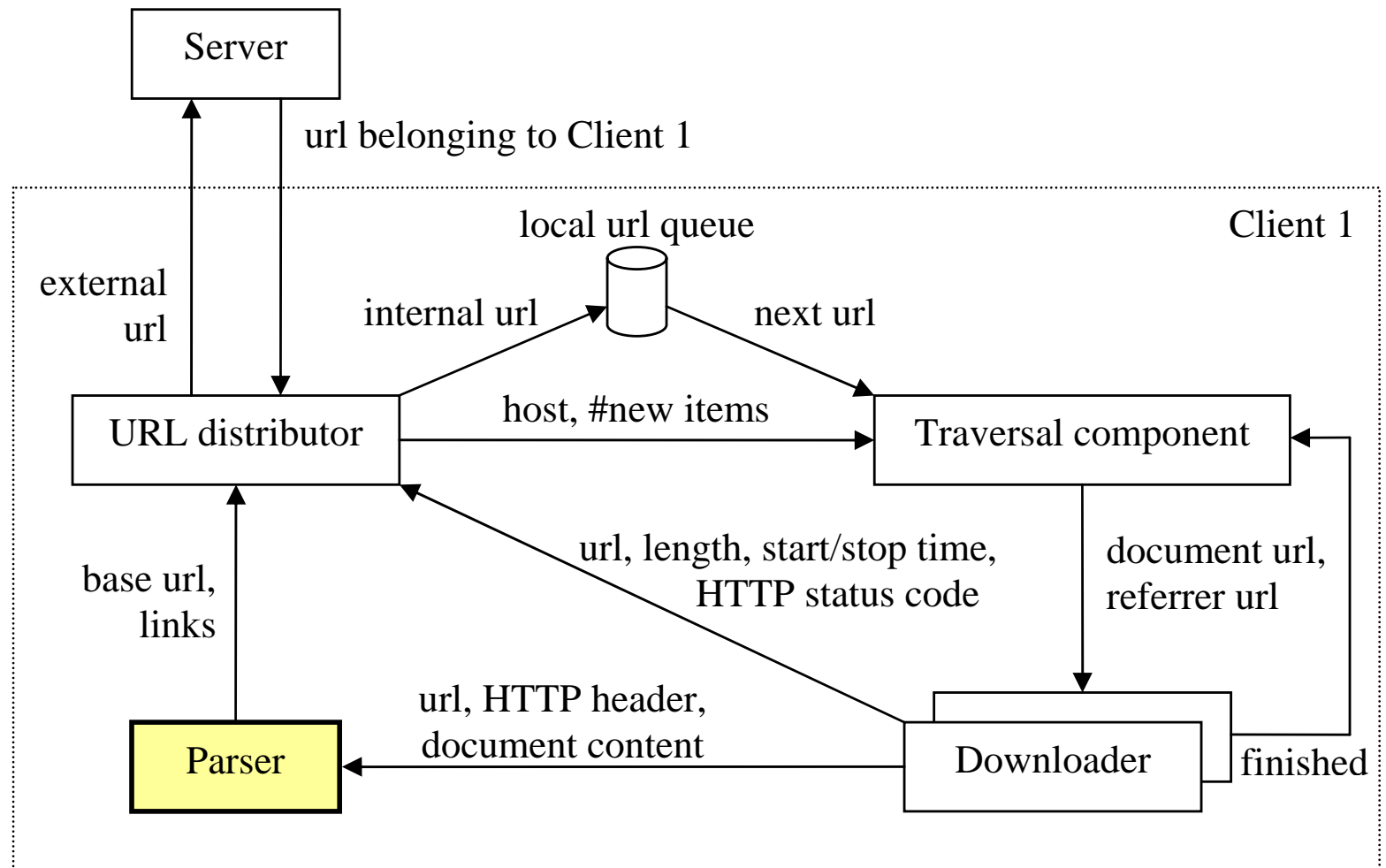
Load balancing

Parsing

URL distributor  
component

Conclusions

# Parser component



Introduction

Component  
framework

Crawler  
application

Architecture

Server  
components

Client  
components

Traversal

Load balancing

Parsing

URL distributor  
component

Conclusions

# Parser component

Introduction

Component  
framework

Crawler  
application

Architecture

Server  
components

Client  
components

Traversal

Load balancing

Parsing

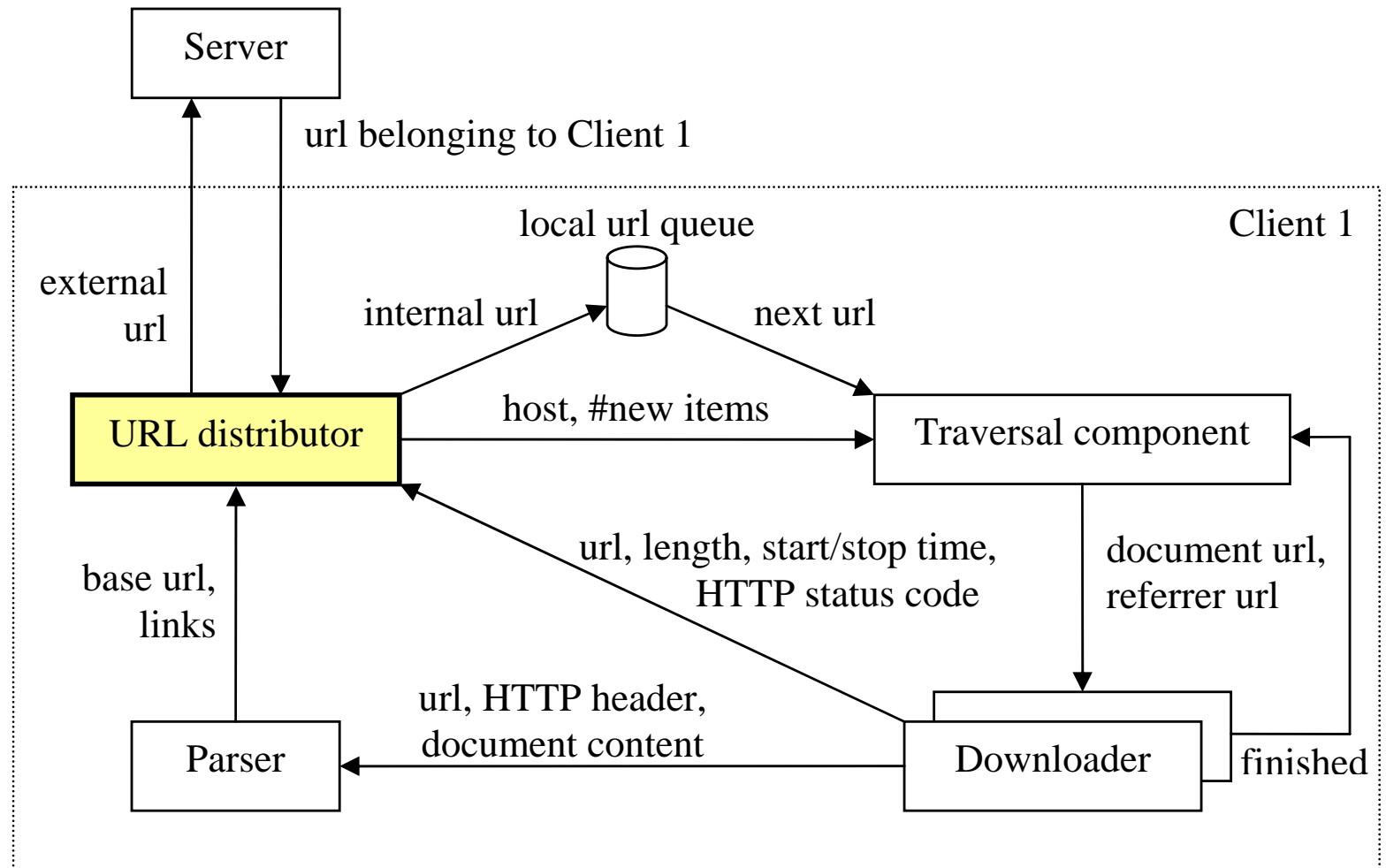
URL distributor  
component

Conclusions

Extracts hyperlinks from documents:

- nonstandard HTML files
- versatile hyperlink manifestations
- performance issues (e.g. interpreting scripts)
- document types and document encoding

# URL distributor component



Introduction

Component  
framework

Crawler  
application

Architecture

Server  
components

Client  
components

Traversal

Load balancing

Parsing

URL distributor  
component

Conclusions

# URL distributor component

Introduction

Component  
framework

Crawler  
application

Architecture

Server  
components

Client  
components

Traversal

Load balancing

Parsing

URL distributor  
component

Conclusions

- configurable URL-patterns:  
what to keep and what to omit
- robot exclusion rules
- crawler traps



Introduction

Component  
framework

Crawler  
application

Conclusions

Future work

Summary

- configuration and monitoring from graphical user interface
- dynamic, run-time re-configuration
- dedicated data structures instead of databases for large sets of data
- crawl on simulated data set and actual crawl

Introduction

Component  
framework

Crawler  
application

Conclusions

Future work

Summary

- component framework for general tasks
- loosely-coupled field-specific components
- open, extensible, scalable architecture
- transparent caching mechanisms
- declarative configurability

Available for download at SourceForge.net

<http://sourceforge.net/projects/webcrawler>

Introduction

Component  
framework

Crawler  
application

Conclusions

Future work

Summary

# Thank you for your attention!