Distributed Searching in Web Information Systems

Krasimir Trichkov

Abstract: *This paper aims to present architecture for search and retrieval in heterogeneous databases (library applications) using ANSI/NISO Z39.50. This standard defines a client/server based service and protocol for Information Retrieval. It specifies procedures and formats for a client to search a database provided by a server, retrieve database records, and perform related information retrieval functions. The protocol addresses communication between information retrieval applications at the client and server. The paper examines the potential of Z39.50 to enable new methods of data creation and exchange in library networks.*

Key words: *ANSI/NISO Z39.50, Internet, Database, Zebra, Zap, Php/Yaz.*

INTRODUCTION

Z39.50 is a computer-to-computer communications protocol designed to support searching and retrieval of information (full-text documents, bibliographic data, images, multimedia) in a distributed network environment. Based on client/server architecture and operating over the Internet, the Z39.50 protocol is supporting an increasing number of applications. And like the dynamic network environment in which it is used, the standard is evolving to meet the changing needs of information creators, providers, and users (for effective information exchange). A large amount of information in an organization leads to the need of making possible to extract the necessary data and to access them everywhere and any time.

DESCRIPTION OF ANSI/NISO Z39.50

ANSI/NISO Z39.50 is the American National Standard Information Retrieval Application Service Definition and Protocol Specification for Open Systems Interconnection. It defines a standard way for two computers to communicate for the purpose of information retrieval. Z39.50 makes it easier to use large information databases by standardizing the procedures and features for searching and retrieving information. Specifically, Z39.50 supports information retrieval in a distributed, client and server environment where a computer operating as a client submits a search request to another computer acting as an information server. Software on the server performs a search on one or more databases and creates a result set of records that meet the criteria of the search request. The server returns records from the result set to the client for processing. The power of Z39.50 is that it separates the user interface on the client side from the information servers, search engines, and databases. Z39.50 provides a consistent view of information from a wide variety of sources, and it offers client implementers the capability to integrate information from a range of databases and servers [1].

ABSTRACT MODEL OF Z39.50 PROTOCOL

Z39.50 recognizes that information retrieval consists of two primary components – selection of information based upon some criteria and retrieval of that information, and it provides a common language for both activities. Z39.50 standardizes the manner in which the client and the server communicate and interoperate even when there are differences between computer systems, search engines, and databases [2].

Z39.50 is an applications-layer protocol originally modelled within the Open Systems Interconnection (OSI) Basic Reference Model developed by the International Organization for Standardization (ISO). Applications-layer protocols support the communications requirements of and interact directly with computer programs residing on clients and servers that perform specific operations.

Figure 1 shows abstract model of Z39.50 client/server architecture [3].

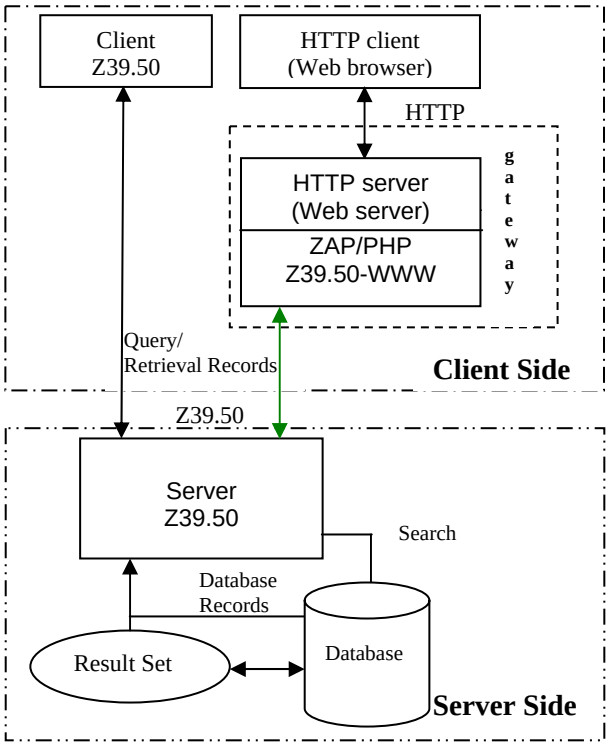

Figure 1 Common client/server architecture

Networked information retrieval requires:

- Identifying a target to search
- A vocabulary for expressing search requests, search criteria, retrieval requests, etc.
- Methods to encode the requests and responses from the target
- Methods to transport the requests and responses across a network

A series of messages passing between the client and server (defined by what the standard calls the Initialization Facility) establish a connection, initiate a Z39.50 session, and negotiate expectations and limitations on the activities that will occur (e.g., maximum size of the records that will be transferred from the server to the client, the version of the protocol supported, etc.). After these agreements are negotiated, the client may submit a query. The Z39.50 client translates the query into a standardized representation and passes it to a Z39.50 server (defined by the Search Facility). The server executes the search against a database(s), and a result set is created. The client can then ask for records from the result set or request from the server additional processing of the result set (defined by the Retrieval Facility). Upon receipt of the records, the client may process the records and display records to the user. The extent to which a client can perform additional processing on retrieved records (e.g., combining records from several separate searches) will depend on the user interface software since it is separate from the Z39.50 client software. The protocol supports Search, Scan, Sort, Extended Services, Explain, Segmentation, Proximity Searching.

PHYSICAL IMPLEMENTATION

ANSI/NISO Z39.50 can be implemented on any platform. This means that Z39.50 enables different computer systems (with different operating systems, hardware, search engines, database management systems) to interoperate and work together seamlessly. A Z39.50 implementation enables one interface to access multiple systems providing end users with nearly transparent access to other systems [4, 5].

Physical implementation of this Z39.50 protocol is based on common client/server architecture (Figure 1). There are two variants for data exchange – with client architecture (using Z39.50 client program) and Web/Z39.50 gateway architecture (using web based interface – browser-based implementation). Web based interface is available on HTTP Server www3.iccs.bas.bg and use also same address for Z39.50 Server (Zebra Server).

A. Database record

Next table (table 1) shows www3.iccs.bas.bg Zebra Server record:

TABLE 1. Zebra Server record

```
<gils>
<Title>State</Title>
<Author>Nadejda Miteva</Author>
<Creator>K.Stoilova, K.Trichkov</Creator>
<purpose>$ 90</purpose>
<URL>Tempera</URL>
<Description>State, 2000</Description>
<Publisher>ICCS</Publisher>
<Contribution>UBA</Contribution>
<Date>2001.10.29</Date>
<Type>Image</Type>
<Format>jpeg</Format>
<Identifier>0095</Identifier>
<Record-source>http://www3.iccs.bas.bg/RecordsUBA/nad_miteva.jpg
</Record-source>
<Language>en</Language>
<Relation>http://www3.iccs.bas.bg/RecordsUBA/nad_miteva.jpg</Relation>
<Coverage>Contemporary Bulgarian Art</Coverage>
<Rights>UBA</Rights>
<Text>Nadejda Miteva,State, 2000</Text>
<DateofLastModification>2001.11.06</DateofLastModification>
<Source>
Nadejda Miteva          nad_miteva.jpg
        Sofia, 1950
        State, 2000
        Tempera, Price:  $ 90
Dimension 374x 524pixels,
resolution 72 pixels/inch
</Source>
<xml>
<name>State</name>
<item_id>94</item_id>
<description>State, 2000</description>
<producer>Nadejda Miteva</producer>
<price>$ 90</price>
<host_ref>http://www3.iccs.bas.bg/RecordsUBA/nad_miteva.jpg</host_ref>
</xml>
</gils>
```

B. Zebra Search Request/Response

Table 2 defines www3.iccs.bas.bg Zebra Server Request/Response model:

TABLE 2. Zebra Server Request/Response model

```
H:\Program Files\Zebra\test\gils>H:\Progra~1\Zebra\bin\zebraidx -t grs.sgml update records
15:47:30-23/12: [log] Zebra version 1.3.15 $Date: 2004/01/15 14:22:22 $
15:47:30-23/12: [log][app2] zebra_start zebra.cfg
15:47:30-23/12: [log][app2] zebra_register_open rw = 1 useshadow=0 p=00514990,n= ,rp=(none)
15:47:30-23/12: [log][app2] updating records
```

```
15:47:30-23/12: [log] dir records
15:47:34-23/12: [log] add grs.sgml records/esdd0090.txt 0
15:47:34-23/12: [log] add grs.sgml records/esdd0091.txt 0
15:47:34-23/12: [log] add grs.sgml records/esdd0092.txt 0
15:47:34-23/12: [log] add grs.sgml records/esdd0093.txt 0
15:47:34-23/12: [log] add grs.sgml records/esdd0094.txt 0
15:47:34-23/12: [log] add grs.sgml records/esdd0095.txt 0
15:47:34-23/12: [log] add grs.sgml records/esdd0096.txt 0
15:47:34-23/12: [log] add grs.sgml records/esdd0097.txt 0
15:47:34-23/12: [log] add grs.sgml records/esdd0098.txt 0
15:47:34-23/12: [log] add grs.sgml records/esdd0099.txt 0
15:47:34-23/12: [log] add grs.sgml records/esdd0100.txt 0
15:47:38-23/12: [log] zebra_end_trans
15:47:38-23/12: [log] sorting section 1
15:47:38-23/12: [log] writing section 1
15:47:38-23/12: [log] finished section 1
15:47:39-23/12: [log] Iterations . . .   2331
15:47:39-23/12: [log] Distinct words .    704
15:47:39-23/12: [log] Updates. . . . .      0
15:47:39-23/12: [log] Deletions. . . .      0
15:47:39-23/12: [log] Insertions . . .    704
15:47:39-23/12: [log][app2] zebra_register_close p=00514990
15:47:39-23/12: [log] Records:     121 i/u/d 121/0/0
15:47:39-23/12: [log][app2] zebra_stop
H:\Program Files\Zebra\test\gils>H:\Progra~1\Zebra\bin\zebrasrv
15:47:39-23/12: [log][app2] zebra_start zebra.cfg
15:47:39-23/12: [log] Starting server zebrasrv
15:47:39-23/12: [log] Adding dynamic Z3950 listener on tcp:@:9999
15:47:39-23/12: [log] Entering event loop.
15:48:30-23/12: [log] Got initRequest
15:48:30-23/12: [log] Id:        YAZ (id=81)
15:48:30-23/12: [log] Name:      PHP/YAZ
15:48:30-23/12: [log] Version:   1.6
15:48:30-23/12: [log] Negotiated to v3: srch prst extendedServices namedresults scan sort
15:48:30-23/12: [log] Got SearchRequest.
15:48:30-23/12: [log] ResultSet 'default'
15:48:30-23/12: [log] Database 'Default'
15:48:30-23/12: [log] RPN query. Type: Bib-1
15:48:30-23/12: [log]  term 'wood' (general)
15:48:30-23/12: [log] ResultSet 'default'
15:48:30-23/12: [log][app2] zebra_register_open rw = 0 useshadow=0 p=0051C5A8,n= ,rp=(none)
15:48:31-23/12: [log] dict_lookup_grep: (wood)
15:48:31-23/12: [log] resultSetRank
15:48:31-23/12: [log] term="wood" nn=0 type=void count=0
15:48:31-23/12: [log] 0 keys, 0 distinct sysnos
15:48:31-23/12: [log] resultCount: 0
15:48:31-23/12: [log] Connection closed by client
15:48:31-23/12: [log][app2] zebra_register_close p=0051C5A8
15:48:43-23/12: [log] Got initRequest
15:48:43-23/12: [log] Id:        YAZ (id=81)
15:48:43-23/12: [log] Name:      PHP/YAZ
15:48:43-23/12: [log] Version:   1.6
15:48:43-23/12: [log] Negotiated to v3: srch prst extendedServices namedresults scan sort
15:49:33-23/12: [log] Got SearchRequest.
15:49:33-23/12: [log] ResultSet 'default'
15:49:33-23/12: [log] Database 'Default'
15:49:33-23/12: [log] RPN query. Type: Bib-1
15:49:33-23/12: [log]  term 'state' (general)
15:49:33-23/12: [log] ResultSet 'default'
15:49:33-23/12: [log][app2] zebra_register_open rw = 0 useshadow=0 p=0051C5A8,n= ,rp=(none)
15:49:33-23/12: [log] dict_lookup_grep: (state)
15:49:33-23/12: [log] resultSetRank
```

```
15:49:33-23/12: [log] term="state" nn=1 type=void count=1
15:49:33-23/12: [log] 1 keys, 1 distinct sysnos
15:49:33-23/12: [log] resultCount: 1
15:49:33-23/12: [log] Request to pack 1+1+default
15:49:33-23/12: [log] Connection closed by client
15:49:33-23/12: [log][app2] zebra_register_close p=0051C5A8
```

C. PHP_YAZ module for Search and Retrieval

PHP source code (table 3) for Search and Retrieval using PHP_YAZ library:

TABLE 3. PHP source code model

```
mysql_connect($host,$user,$userpass) or die("Connect failed");
        if(mysql_select_db($base)==FALSE)
          { mysql_create_db($base) or die("DataBase '$base' select failed<br>"); }
        $query="select url from targets";
        $result=mysql_query($query) or die("Error in query");
        $i=0;
        while ($row=mysql_fetch_row($result))
            {
              $hosts[$i]=$row[0];
              $i++;
            }
        $num=count($hosts);
        $fb="";
        for ($i=0; $i<$num; $i++)
            {
              $id[$i] = yaz_connect($hosts[$i]);
           yaz_syntax($id[$i],"xml");
           yaz_range($id[$i],$starting,$hit);
              if ($fb=="full")
            yaz_element ($id[$i], "f");
           else
              yaz_element ($id[$i], "f");
              $string=split(" ",$term);
              if (isset($string[1]))
              {
                      $term1="@or ".$term;
              }
              else
              {
                      $term1=$term;
              }
                 //$term1="@set ".$term;
            //echo $term1;
              yaz_search($id[$i],"rpn",$term1);
           yaz_wait();
           $error = yaz_error($id[$i]);
           if (!empty($error)) {
           echo "<br><b>$hosts[$i]</b> - Error: $error";
              }
              else
              {
             $hits=yaz_hits($id[$i]);
             $hitsumm+=$hits;
           echo "<br><b>$hosts[$i]</b> - Result Count: $hits";
           }
            }
```

D. Software components

Zebra - Zebra is a fielded free-text indexing and retrieval engine with a Z39.50 fronted. Zebra is a high-performance, general-purpose structured text indexing and retrieval engine. It reads structured records in a variety of input formats (eg. email, XML, MARC. Zebra supports large databases (more than ten gigabytes of data, tens of millions of records). It supports incremental, safe database updates on live systems.

ZAP - ZAP is a module (to Web servers), which allows you to build simple WWW interfaces to Z39.50 servers. ZAP hides most of the complexity of session management, parallel searching. The integration of system into the popular Web servers offers several advantages to the operators and users of the software, including simplified maintenance of the Module, and improved performance.

PHP/YAZ - This extension offers a PHP interface to the YAZ toolkit that implements the Z39.50 protocol for Information Retrieval. With this extension its easily to implement a Z39.50 origin (client) that searches or scans Z39.50 targets (servers) in parallel.

This is free software [6] that can work on various operating systems and various Web Servers.

CONCLUSION AND FUTURE WORK

The architecture for search and retrieval in heterogeneous databases (library applications) using ANSI/NISO Z39.50 were presented. Definite is the potential of Z39.50 to enable new methods of data creation and exchange in library networks. Definitely are software components of the protocol. The protocol is platform and software independent. As a future work is the problem for optimization of developed searching services. The software implementation can be reached at http://www3.iccs.bas.bg.

REFERENCE

[1] http://www.cni.org

[2] Ivanova E., Application of Distributed Search in Databases for Web Services, International conference ICEST'03, p.291-294

[3] Trichkov Kr. Search and Retrieval Web Services in the Web Information Systems. Proceedings of the International Conference "Automatics and Informatics'04", Bulgaria, Sofia, October, 6-8, 2004, p.5-8.

[4] Stoilov T. E-Centre the European Open Soft-development Multi-platform. International workshop "Next Generation of Open Development Platform for Software and Services, 21 July, 2003, Vilnius, Lithuania, p.1-9.

[5] Tsenov *M.,* Data Exchange for Distributed Network Systems. Preprints of the International IFAC workshop DECOM-TT 2004, Bansko, Bulgaria, p. 239

[6] http://www.indexdata.dk

ABOUT THE AUTHOR

Assistant Prof. M.Sc. Eng. Krasimir Trichkov, Institute of Computer and Communication Systems – Bulgarian Academy of Sciences, Acad. G. Bonchev bl.2, 1113 Sofia, Bulgaria, E-mail: krasi@hsi.iccs.bas.bg