3.5 SSL Verbindungen

- JSSE (Java Secure Socket Extension)
- `import javax.net.ssl.*`
- Wesentliche Änderung im **Client** Programm: (Factory Pattern)

  Ersetze

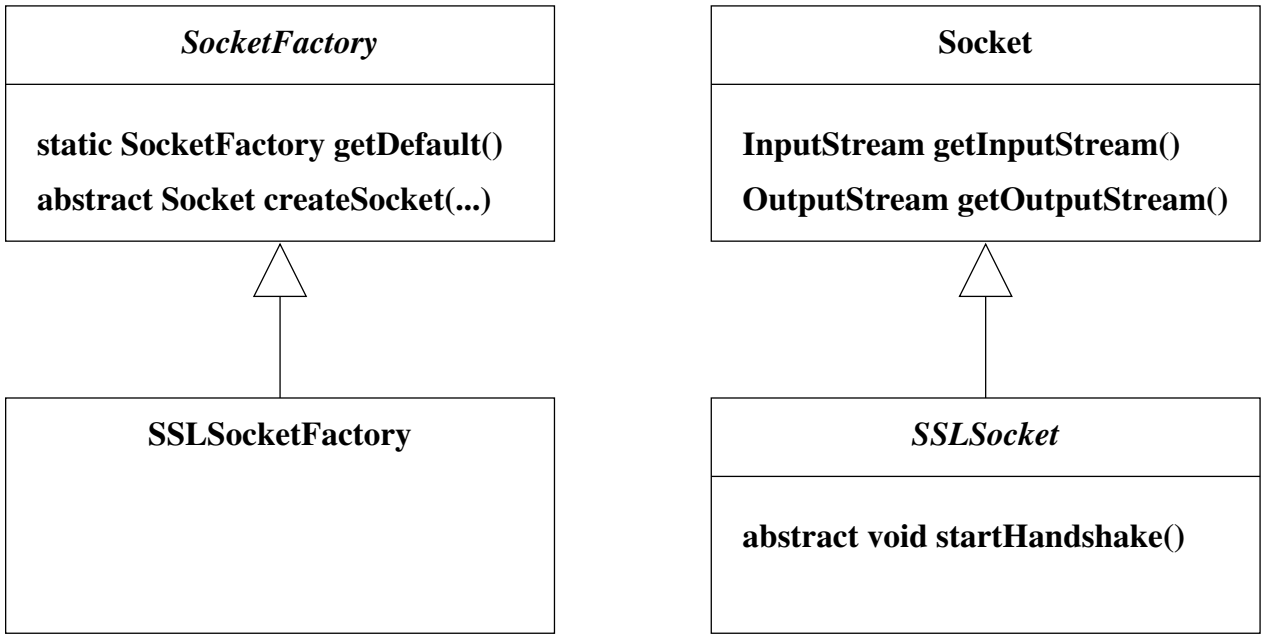
```
Socket s = new Socket (hostname, portnumber)
```

  durch

```
SSLSocketFactory sf = (SSLSocketFactory)SSLSocketFactory.getDefault();
SSLSocket s = (SSLSocket)sf.createSocket(hostname, portnumber);
```

- Dafür muss SSL konfiguriert sein (siehe unten).

SSLSocketFactory



SSL Server Sockets

- Wesentliche Änderung im **Server** Programm:

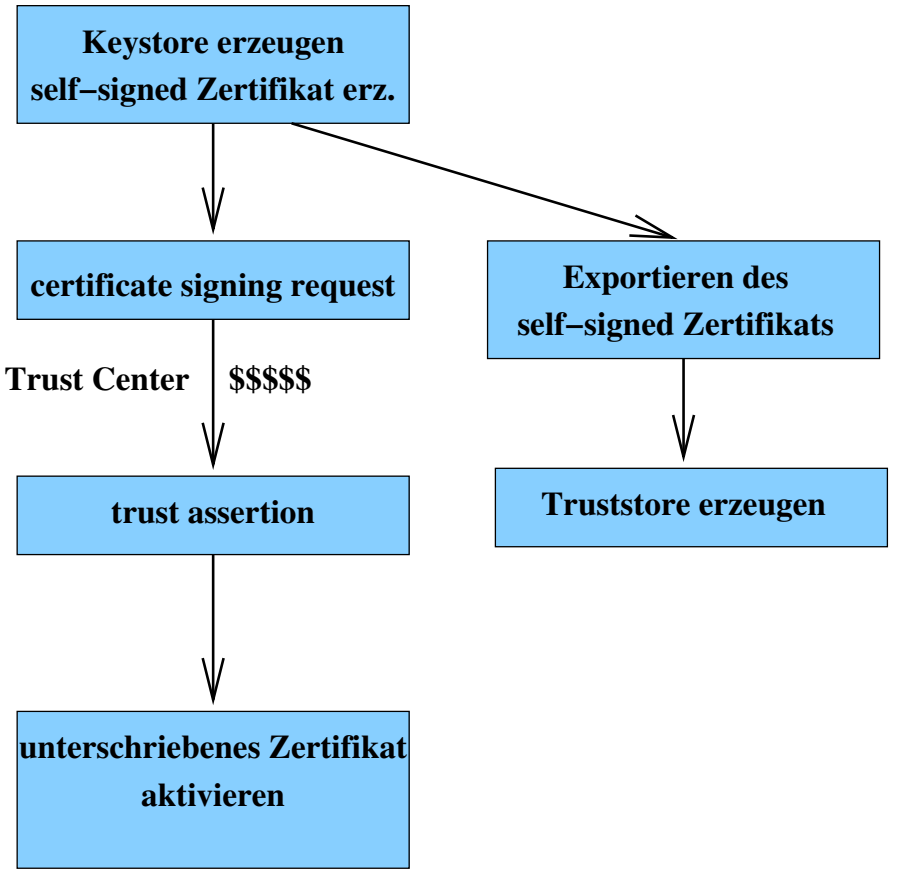
  Ersetze

```
ServerSocket ss = new ServerSocket (portnumber)
// ...
Socket con = ss.accept ()
```

  durch

```
SSLServerSocketFactory sf =
    (SSLServerSocketFactory)SSLServerSocketFactory.getDefault();
SSLServerSocket ss =
    (SSLServerSocket)sf.createServerSocket(portnumber);
// ...
SSLSocket con = (SSLSocket)ss.accept();
```

Konfiguration von SSL: Zertifikate



Erzeugen des Key Store

```
> keytool -genkey -alias widgetorg -keyalg RSA -validity 7 -keystore serverkey.jks
Enter keystore password:  ToPsEcReT
What is your first and last name?
  [Unknown]:  www.widget.inc
What is the name of your organizational unit?
  [Unknown]:  Web Division
What is the name of your organization?
  [Unknown]:  Widget Inc.
What is the name of your City or Locality?
  [Unknown]:  Punxsutawney
What is the name of your State or Province?
  [Unknown]:  Pennsylvania
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=www.widget.inc, OU=Web Division, O=Widget Inc., L=Punxsutawney, ST=Pennsylvania, C=US correct?
  [no]:  yes

Enter key password for <widgetorg>
(RETURN if same as keystore password):
```

- Schlüsselpaar, Self-Signed Zertifikat in *serverkey.jks*

Certificate Signing Request

```
> keytool -certreq -alias widgetorg -keystore serverkey.jks -keyalg RSA -file widgetorg.csr
Enter keystore password:  ToPsEcReT
```

- Einsenden von *widgetorg.csr* an certificate authority
- Zurück: unterschriebenes Zertifikat *widgetorg.cer*, Zertifikat der certificate authority *ca.cer*

Trust Assertion

```
> keytool -import -alias root -keystore serverkey.jks -trustcacerts -file ca.cer
Enter keystore password: ToPsEcReT
Certificate was added to keystore
```

- Dieser Schritt kann übersprungen werden, falls certificate authority bereits bekannt

Zertifikat aktivieren

```
> keytool -import -alias widgetorg -keystore serverkey.jks -file widgetorg.cer
Enter keystore password: ToPsEcReT
Certificate was added to keystore
```

Zertifikat verwenden

```
> java -Djavax.net.ssl.keyStore=serverkey.jks
       -Djavax.net.ssl.keyStorePassword=ToPsEcReT
       MySecureServer 8443
```

- Falls certificate authority allgemein bekannt, kann jeder Client eine Verbindung herstellen

Alternative: self-signed Zertifikat akzeptieren

- Exportiere das self-signed Zertifikat nach Konstruktion des key stores:

```
> keytool -export -keystore serverkey.jks -alias widgetorg -file widgetorgself.cer
Enter keystore password:  ToPsEcReT
Certificate stored in file <widgetorgself.cer>
```

- Erzeuge dann einen *trust store*, der alle vertrauenswürdigen Schlüssel enthält.

```
> keytool -import -alias widgetorg -file widgetorgself.cer -keystore truststore.jks
Enter keystore password:  HuShHuSh
Owner: CN=www.widget.inc, OU=Web Division, O=Widget Inc., L=Punxsutawney, ST=Pennsylvania, C=US
Issuer: CN=www.widget.inc, OU=Web Division, O=Widget Inc., L=Punxsutawney, ST=Pennsylvania, C=US
Serial number: 42665235
Valid from: Wed Apr 20 14:59:33 CEST 2005 until: Wed Apr 27 14:59:33 CEST 2005
Certificate fingerprints:
 MD5:  4D:A7:09:CB:1A:8E:5F:91:5E:7A:2F:F1:CD:16:B6:4F
 SHA1: C6:90:05:6D:1D:B8:B1:5D:C9:83:BF:9F:79:2C:FD:28:54:58:B9:D6
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

- Verwendung in einem Client mit

```
> java -Djavax.net.ssl.trustStore=truststore.jks
       -Djavax.net.ssl.trustStorePassword=HuShHuSh
       MySecureClient 8443
```

Wenn was schief geht ...

```
import java.net.*;
import java.io.*;
import javax.net.ssl.*;
import java.security.cert.*;

class CertificationPath {
  public static void main (String[] args) {
    try {
      // Create the client socket
      int port = 443;
      String hostname = args[0];
      SSLSocketFactory factory = HttpsURLConnection.getDefaultSSLSocketFactory();
      SSLSocket socket = (SSLSocket)factory.createSocket(hostname, port);

      // Connect to the server
      socket.startHandshake();
```

```
      // Retrieve the server's certificate chain
      java.security.cert.Certificate[] serverCerts =
        socket.getSession().getPeerCertificates();

      for (int i=0; i<serverCerts.length; i++) {
        System.out.println("Server certificate type: "+serverCerts[i].getType());
        if (serverCerts[i] instanceof X509Certificate) {
          X509Certificate c = (X509Certificate)serverCerts[i];
          System.out.println(" Subject: "+c.getSubjectDN());
          System.out.println(" Issuer: "+c.getIssuerDN());
        }
      }

      // Close the socket
      socket.close();
    } catch (SSLPeerUnverifiedException e) {
    } catch (IOException e) {
    }
  }
}
```


3.6 UDP Sockets

- Wichtig: UDP Ports $\neq$ TCP Ports
- Java API: Zwei Klassen
  - `DatagramPacket` repräsentiert ein Datenpaket (zum Versenden oder nach dem Empfang)
  - `DatagramSocket` repräsentiert die eigentliche Verbindung

Klasse `java.net.DatagramPacket`

nur Aufbau von Datenstruktur, keine Verbindung!

Wichtige Konstruktoren

- ```
  DatagramPacket(byte[] buf, int length)
  ```
  zum Empfang von `length` Bytes in `buf`
- ```
  DatagramPacket(byte[] buf, int length, InetAddress address, int port)
  ```
  vorbereitet zum Versenden von `length` Bytes aus `buf` an `address` und `port`
  Beachte: die Adresse des Ziels befindet sich im Paket!

Wichtige Methoden `java.nat.DatagramPacket`

Empfangen

- `byte[] getData()`
- `int getLength()`
- `InetAddress getAddress()`
- `int getPort()`

Senden

- `void setData(byte[] buf)`
- `void setLength(int length)`
- `void setAddress(InetAddress iaddr)`
- `void setPort(int iport)`

Klasse `java.net.DatagramSocket`

Wichtige Konstruktoren

- `DatagramSocket()`
- `DatagramSocket(int port)`

Wichtige Methoden

- `void send(DatagramPacket p) throws IOException`
- `void receive(DatagramPacket p) throws IOException`
- `void close()`

Ablauf

Senden

```
s = new DatagramSocket ();
p = new DatagramPacket (b,l);
p.setAddress (...);
p.setPort (...);
p.setData (...);
s.send (p);
```

Empfangen

```
s = new DatagramSocket (myport);
p = new DatagramPacket (b,l);
s.receive (p);
result = p.getData ();
sender = p.getAddress ();
seport = p.getPort ();
```

Beispiel — ein Client für `daytime` RFC 867

```
public class Daytime {
    static final int BUFSIZE = 128;
    static final int DAYTIME = 13;      // portnumber of daytime service
    //...
    public static String getTime (String hostname)
        throws Exception {
        byte[] buffer = new byte[BUFSIZE];
        InetAddress server = InetAddress.getByName (hostname);
        DatagramPacket answer = new DatagramPacket (buffer, BUFSIZE);
        DatagramSocket s = new DatagramSocket ();
        answer.setAddress (server);
        answer.setPort (DAYTIME);
        s.send (answer);        // contents do not matter
        s.receive (answer);
        s.close ();
        int len = answer.getLength ();
        buffer = answer.getData ();
        while (buffer[len-1] == 10 || buffer[len-1] == 13) {
            len--;
        }
        return new String (buffer, 0, len);
    }
}
```

Beispiel — ein Server für daytime RFC 867

```
public class DaytimeServer {

    static final int BUFSIZE = 128;
    static final int DAYTIME = 13;      // portnumber for daytime service
    // ...
    public static void serveTime (int port)
        throws Exception {
        byte[] buffer = new byte[BUFSIZE];
        DatagramPacket p = new DatagramPacket (buffer, BUFSIZE);
        DatagramSocket s = new DatagramSocket (port);
        // while (true) {
        s.receive (p);          // contents do not matter
        Date d = new GregorianCalendar ().getTime ();
        System.out.println ("Sending: " + d);
        String answer = d.toString ();
        p.setData ((answer + "\r\n").getBytes ());
        p.setLength (answer.length () + 2);
        s.send (p);
        // }
        s.close ();
    }
}
```

3.7 UDP vs. TCP

Application	Application-layer protocol	Underlying Transport Protocol
electronic mail	SMTP	TCP
remote terminal access	Telnet	TCP
Web	HTTP	TCP
file transfer	FTP	TCP
remote file server	NFS	typically UDP
streaming multimedia	proprietary	typically UDP
Internet telephony	proprietary	typically UDP
Network Management	SNMP	typically UDP
Routing Protocol	RIP	typically UDP
Name Translation	DNS	typically UDP

3.8 DNS, ein Paket-Protokoll

Hintergrund: RFC 1034. Technische Beschreibung: RFC 1035

DNS: Abbildung von *Domainnamen* auf *Resource Records* (RR)

Ein Domainname ist

- Folge von Strings (Labels), getrennt durch und beendet mit "."
- Maximale Länge eines Labels: 63
- Maximale Länge eines Domainnamen: 255 (inkl. der Punkte)

Menge der Domainnamen ist Hierarchie mit Wurzel "."

```
                                                .
                                              de.
                                    uni-freiburg.de.
                          informatik.uni-freiburg.de.
```

Typen von Resource Records (Ausschnitt):

`A`	host address
`NS`	authoritative name server
`CNAME`	canonical name for an alias
`SOA`	zone of authority
`PTR`	domain name pointer
`MX`	mail exchanger

Grundidee

DNS ist verteilte Datenbank, in der jeder Server zuständig (authoritativ) für eine bestimmte Domain ist.

- Abfrage der Datenbank: UDP Nachricht an *beliebigen* Server.
- Abgleich zwischen den Servern: TCP Verbindungen.

3.8.1 Beispielsitzung

`nslookup` ist ein textuelles Werkzeug für DNS-Anfragen, kontaktiert Port `domain` (53) mit UDP

```
shell> /usr/sbin/nslookup -
Default Server:  atlas.informatik.uni-freiburg.de
Address:  132.230.150.3
```

Alle folgenden Fragen beziehen sich auf Address RRs:

```
> set q=a
```

```
> www.informatik.uni-freiburg.de.
Server:  atlas.informatik.uni-freiburg.de
Address:  132.230.150.3

Name:    falcon.informatik.uni-freiburg.de
Address:  132.230.167.230
Aliases:  www.informatik.uni-freiburg.de
```


Frage nach Nameserver RRs:

```
unix> nslookup -
> set q=ns
> informatik.uni-freiburg.de.
Server:  atlas.informatik.uni-freiburg.de
Address:  132.230.150.3

informatik.uni-freiburg.de      nameserver = dns1.fun.uni-freiburg.de
informatik.uni-freiburg.de      nameserver = tolkien.imtek.uni-freiburg.de
informatik.uni-freiburg.de      nameserver = atlas.informatik.uni-freiburg.de
informatik.uni-freiburg.de      nameserver = dns0.fun.uni-freiburg.de
dns1.fun.uni-freiburg.de        internet address = 132.230.200.201
tolkien.imtek.uni-freiburg.de   internet address = 132.230.168.1
atlas.informatik.uni-freiburg.de        internet address = 132.230.150.3
dns0.fun.uni-freiburg.de        internet address = 132.230.200.200
```

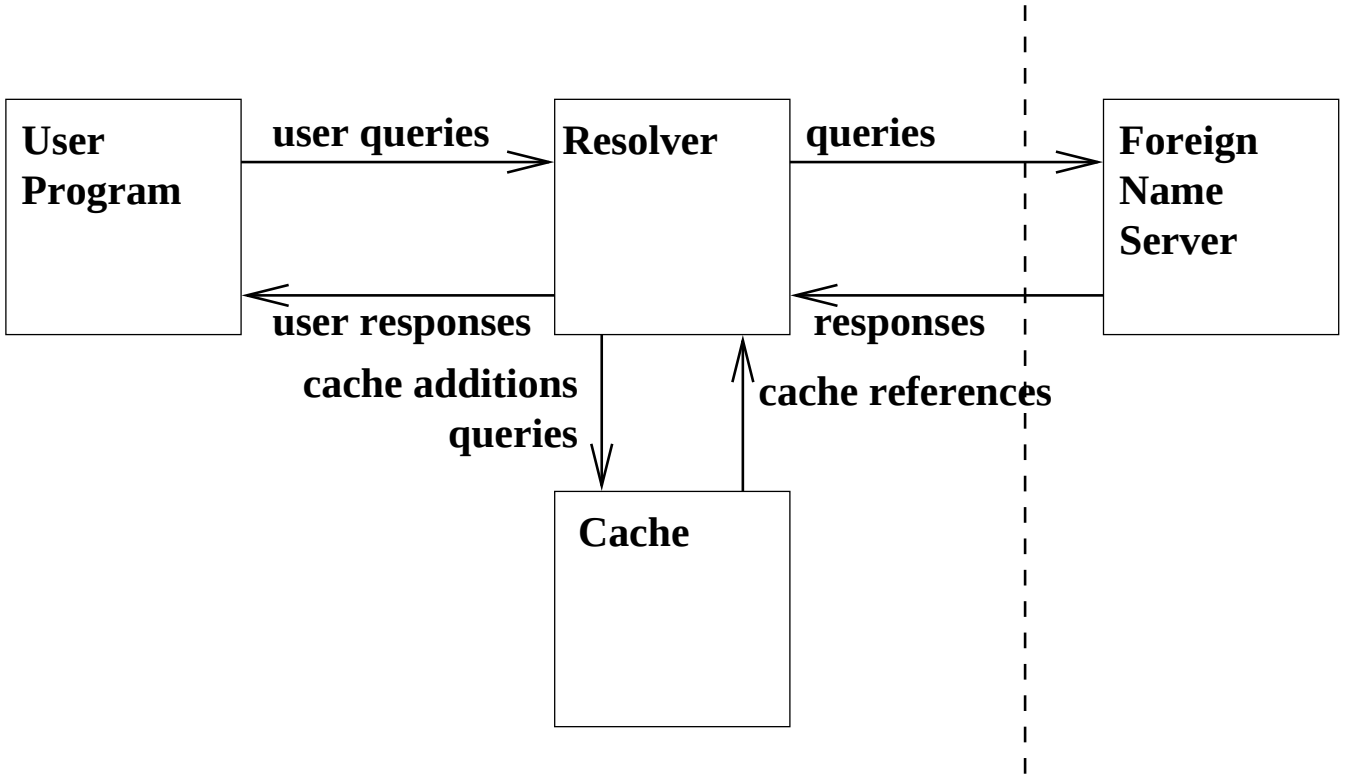
Für Deutschland:

```
> de.
Server:  atlas.informatik.uni-freiburg.de
Address:  132.230.150.3

Non-authoritative answer:
de      nameserver = s.de.net.
de      nameserver = z.nic.de.
de      nameserver = a.nic.de.
de      nameserver = c.de.net.
de      nameserver = f.nic.de.
de      nameserver = l.de.net.

Authoritative answers can be found from:
s.de.net        internet address = 193.159.170.149
z.nic.de        has AAAA address 2001:628:453:4905::53
z.nic.de        internet address = 194.246.96.1
a.nic.de        internet address = 193.0.7.3
c.de.net        internet address = 208.48.81.43
f.nic.de        internet address = 81.91.161.4
f.nic.de        has AAAA address 2001:608:6::5
l.de.net        internet address = 217.51.137.213
```

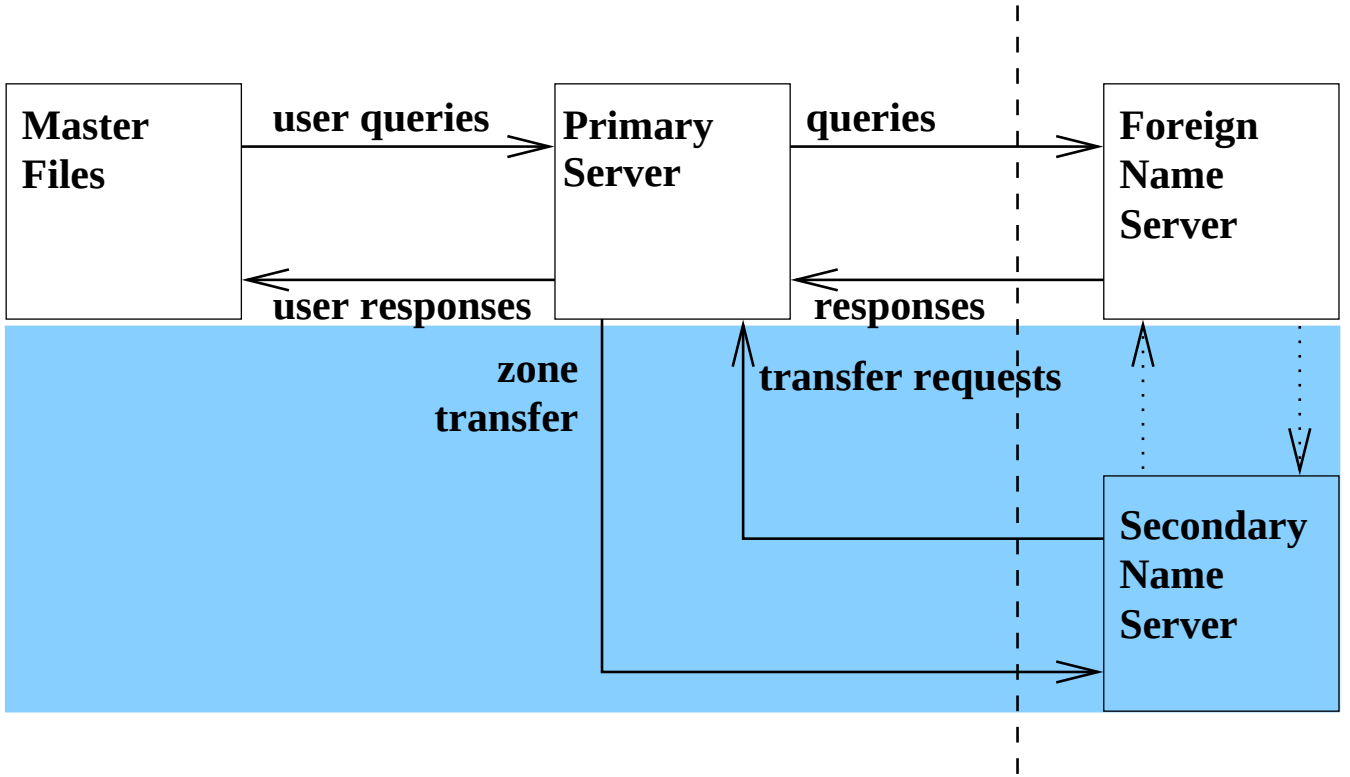
Reverse Query (IP-Adresse → Domainname):

```
> set q=ptr
> 134.2.12.1
Server:  atlas.informatik.uni-freiburg.de
Address:  132.230.150.3

1.12.2.134.in-addr.arpa name = willi.Informatik.Uni-Tuebingen.De
12.2.134.in-addr.arpa   nameserver = dns1.belwue.De
12.2.134.in-addr.arpa   nameserver = dns1.Uni-Tuebingen.De
12.2.134.in-addr.arpa   nameserver = dns3.belwue.De
12.2.134.in-addr.arpa   nameserver = mx01.Uni-Tuebingen.De
12.2.134.in-addr.arpa   nameserver = macon.Informatik.Uni-Tuebingen.De
12.2.134.in-addr.arpa   nameserver = snoopy.Informatik.Uni-Tuebingen.De
dns1.belwue.De  internet address = 129.143.2.1
dns1.Uni-Tuebingen.De   internet address = 134.2.200.1
dns3.belwue.De  internet address = 131.246.119.18
mx01.Uni-Tuebingen.De   internet address = 134.2.3.11
macon.Informatik.Uni-Tuebingen.De       internet address = 134.2.12.17
snoopy.Informatik.Uni-Tuebingen.De      internet address = 134.2.14.4
```

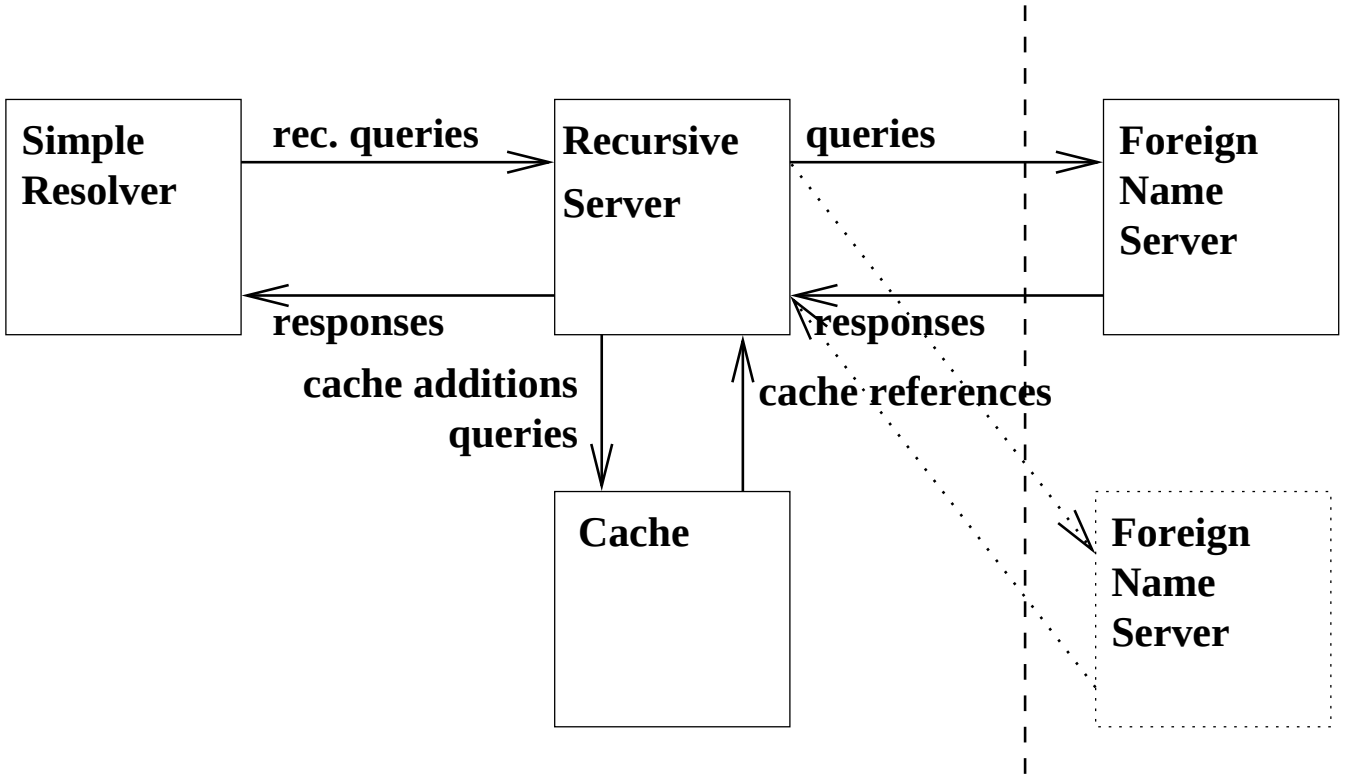
3.8.2 Benutzerperspektive



3.8.3 Primary und Secondary Server



3.8.4 Recursive Queries



3.8.5 Format eines Domainnamens

Folge von Strings (Labels), getrennt durch und beendet mit "."

Maximale Länge eines Labels: 63

Maximale Länge eines Domainnamen: 255 (inkl. der Punkte)

Interne Darstellung: Ein Oktet Länge des Labels, gefolgt von den Zeichen des Labels, wiederholt bis Nulloktet (Label der Länge Null)

Beispiel: `informatik.uni-freiburg.de`

$$\lceil 10 \rceil \texttt{informatik} \lceil 12 \rceil \texttt{uni} - \texttt{freiburg} \lceil 2 \rceil \texttt{de} \lceil 0 \rceil$$

3.8.6 Internes Format eines Resource Record

Feldname	Größe/Oktetts	Beschreibung
NAME	2n	Domainname für den das Record gilt
TYPE	2	Kode für TYPE
CLASS	2	Kode für CLASS
TTL	4	Time to Live, Gültigkeitsdauer/Sek.
RDLENGTH	2	Anzahl der Oktetts im RDATA Feld
RDATA	2d	Inhalt je nach TYPE und CLASS

Kodes für TYPE (Ausschnitt)		
A	1	Host Address
NS	2	authoritative name server
CNAME	5	canonical name for an alias
SOA	6	zone of authority
PTR	12	domain name pointer
MX	15	mail exchanger

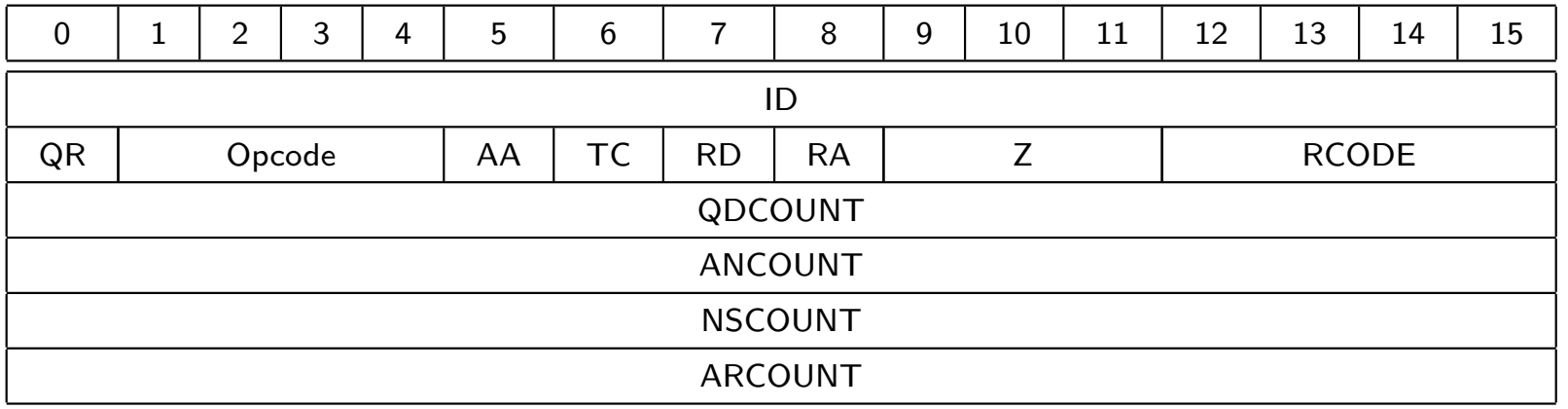
Kodes für CLASS (Ausschnitt)		
IN	1	Internet

3.8.7 Format einer Nachricht

`Header`	
`Question`	Anfrage an den Name-Server
`Answer`	Antworten des Servers
`Authority`	Zeiger auf autorisierten Name-Server
`Additional`	weitere Information

- `Header` immer vorhanden
- `Answer`, `Authority` und `Additional` enthalten je eine Liste von *Resource Records* (RR)
- maximale Länge 512 Oktetts

Header 12 Oktette mit folgendem Inhalt



wobei

ID identifier erzeugt vom Client

QR 0= Frage, 1= Antwort

Opcode Art der Anfrage

- **0=** Standard-Anfrage (QUERY)
- **1=** Inverse Anfrage (IQUERY)
- **2=** Status-Anfrage (STATUS)
- **3–15** reserviert

AA 1= Authoritative Answer

TC 1= Truncated (abgeschnitten)

RD 1= Recursion Desired (Wunsch vom Client)

RA 1= Recursion Available (Anzeige vom Server)

Z immer 0

RCODE Response Code

- **0** kein Fehler
- **1** Formatfehler
- **2** Serverfehler
- **3** Gesuchter Name existiert nicht (nur falls AA)
- **4** nicht implementiert
- **5** Anfrage abgelehnt
- **6–15** reserviert

QDCOUNT Anzahl der Einträge in `Question`

ANCOUNT Anzahl der Resource Records in `Answer`

NSCOUNT Anzahl der Name-Server Resource Records in `Authority`

ARCOUNT Anzahl der Resources Records in `Additional`