DNS/DNSSEC loose ends

Karst Koymans & Niels Sijm

Informatics Institute

University of Amsterdam

Friday, September 21, 2012

1. Encoding of domain names
2. Wildcards in DNS
3. Denial of existence
   - Ignoring wildcards
   - Including wildcard processing

Outline

Composition of domain names

- Domain name is a sequence of labels
- Start at leaf
  1. www
  2. os3
  3. nl
- Start at root
  1. nl
  2. os3
  3. www
- DNS starts at leaf (least significant label)

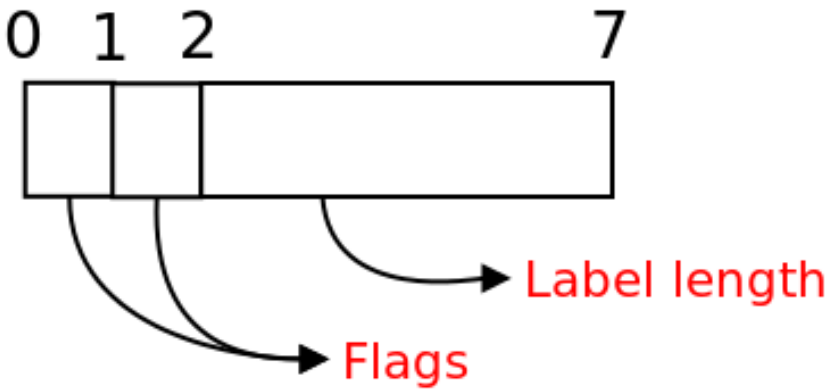
Encoding of domain names

1. Use a delimiter to seperate labels
   - <label> <delimiter> <label>
   - www . os3 . nl .
2. Specify length of labels in label encoding
   - <label> <label>
   - {3,www} {3,os3} {2,nl} {0,}
3. DNS uses latter way of encoding

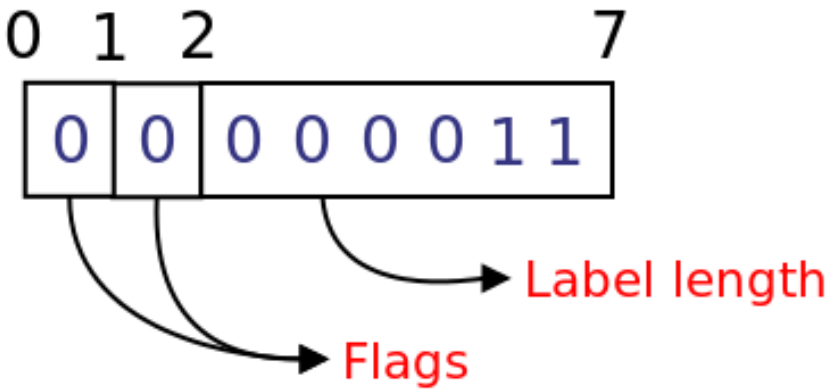
"Normal label length" encoding

- First byte used for length
  - First 2 bits are flags
    - 00 means "normal label length"
  - Remaining 6 specify label length
    - Hence the maximum label length of $2^6 - 1 = 63$ octets
- Remaining bytes contain the label itself
- Number of remaining bytes is encoded in first byte of label

“Normal label length” encoding



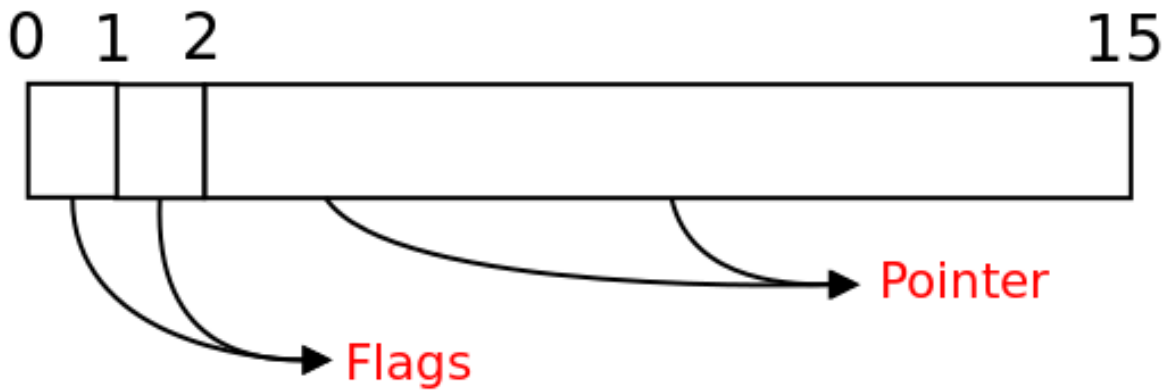
“Normal label length” encoding



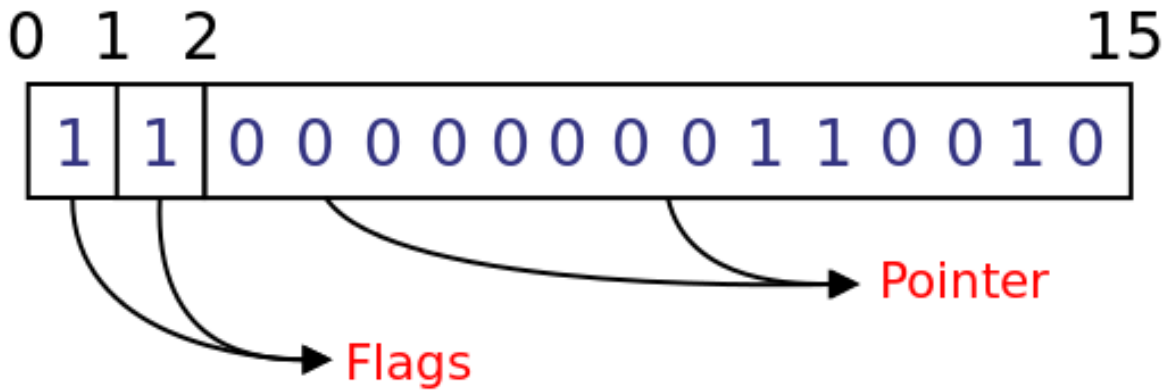
Compressed encoding

- Domain name with compressed encoding has fixed length of 2 bytes
  - First 2 bits are flags
    - 11 means “compressed label”
  - Remaining 6 bits + 8 subsequent bits are used as pointer
    - Points to label at other position in packet
    - Value is offset from beginning of packet
- Saves space when a domain name is used more than once

Compressed encoding



Compressed encoding



Reading domain names

Read first 2 bits of domain name field:

- If values is 00 (normal label length):
  1. If label length is 0 (empty label, thus root):
     - Return sequence of noted labels as domain name
  2. Read 6 subsequent bits and determine length of label
  3. Read first 2 bits of next byte and iterate
- If value is 11 (compressed encoding):
  1. Read 14 subsequent bits and determine position of domain name
  2. Jump to position and decode domain name

One more type: extended

- First byte value: 01000001
  - First two bits are 01
- Defines the use of EDSN0
- Can be used for binary labels
  - IPv6 PTR resource records

Outline

Wildcards according to RFC 1034

From RFC 1034, section 4.3.3

The contents of the wildcard RRs follows the usual rules and formats for RRs. The wildcards in the zone have an owner name that controls the query names they will match. The owner name of the wildcard RRs is of the form "*.<anydomain>", where <anydomain> is any domain name. <anydomain> should not contain other * labels, and should be in the authoritative data of the zone. The wildcards potentially apply to descendants of <anydomain>, but not to <anydomain> itself. Another way to look at this is that the "*" label always matches at least one whole label and sometimes more, but always whole labels.

Wildcards according to RFC 1034

From RFC 1034, section 4.3.3

Wildcard RRs do not apply:

- When the query is in another zone. That is, delegation cancels the wildcard defaults.
- When the query name or a name between the wildcard domain and the query name is known to exist. For example, if a wildcard RR has an owner name of "*.X", and the zone also contains RRs attached to B.X, the wildcards would apply to queries for name Z.X (presuming there is no explicit information for Z.X), but not to B.X, A.B.X, or X.

Problems with wildcards in RFC 1034

- Notions are intuitive, not well-defined
  - When does a domain name "exist"?
  - How does matching work exactly?
  - What about empty non-terminals?
- RFC 4592 tries to clarify all of this
  - Defines "existence of a domain name"
  - Defines "asterisk label" and "wildcard domain name"
  - Defines "source of synthesis" and "closest encloser"

Wildcard supporting definitions

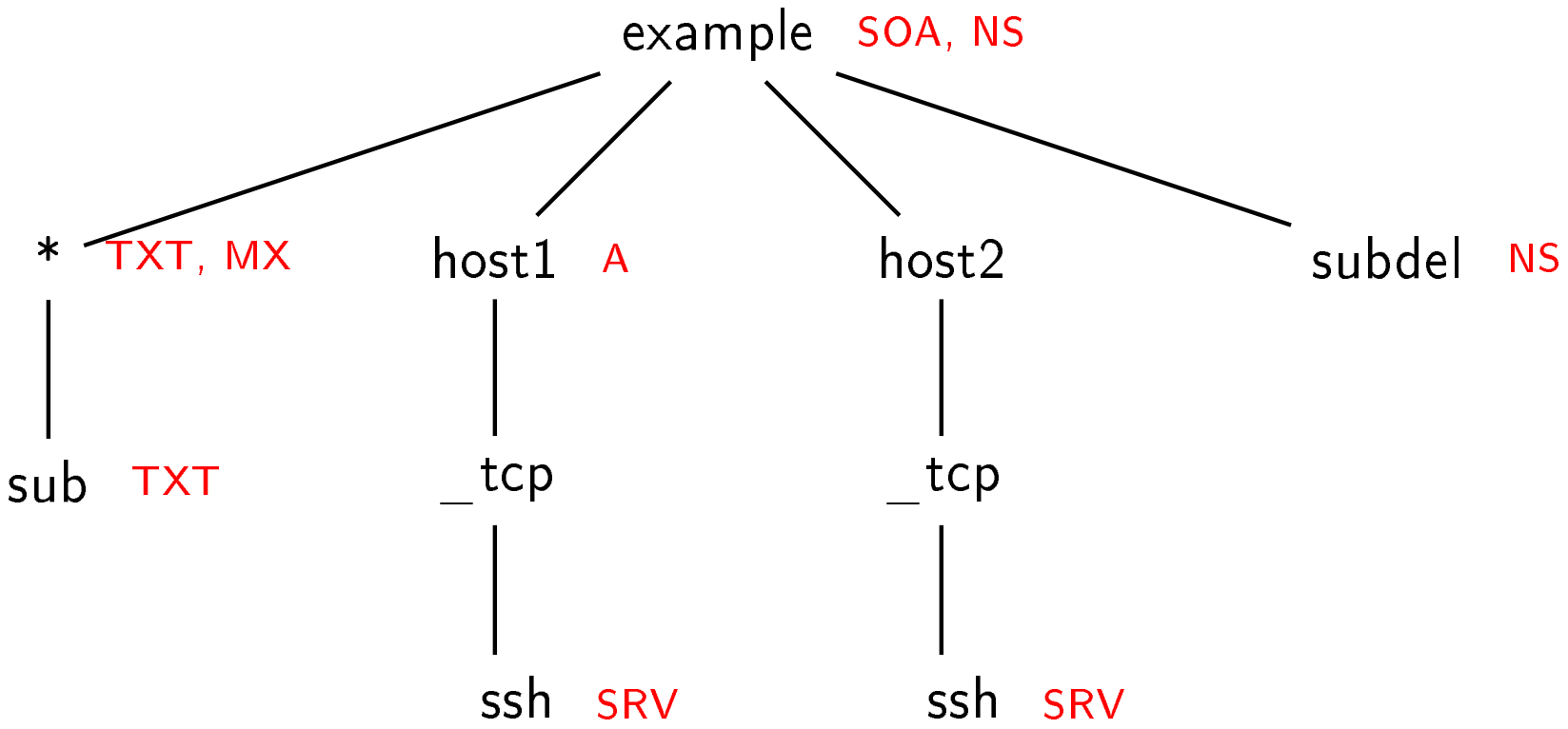
Definitions

- A domain name exists if itself or
  any of its descendants has at least one RR
  - In particular empty non-terminals exist
- An asterisk label is a label of length 1
  containing as only octet the ASCII equivalent of "*"
- A wildcard domain name is a domain name
  with an asterisk label as its leftmost label
- The closest encloser of a query name
  is the longest matching ancestor that exists
- The source of synthesis of a query name
  is the domain name "*.<closest encloser>" (if it exists)

RFC 4592 example

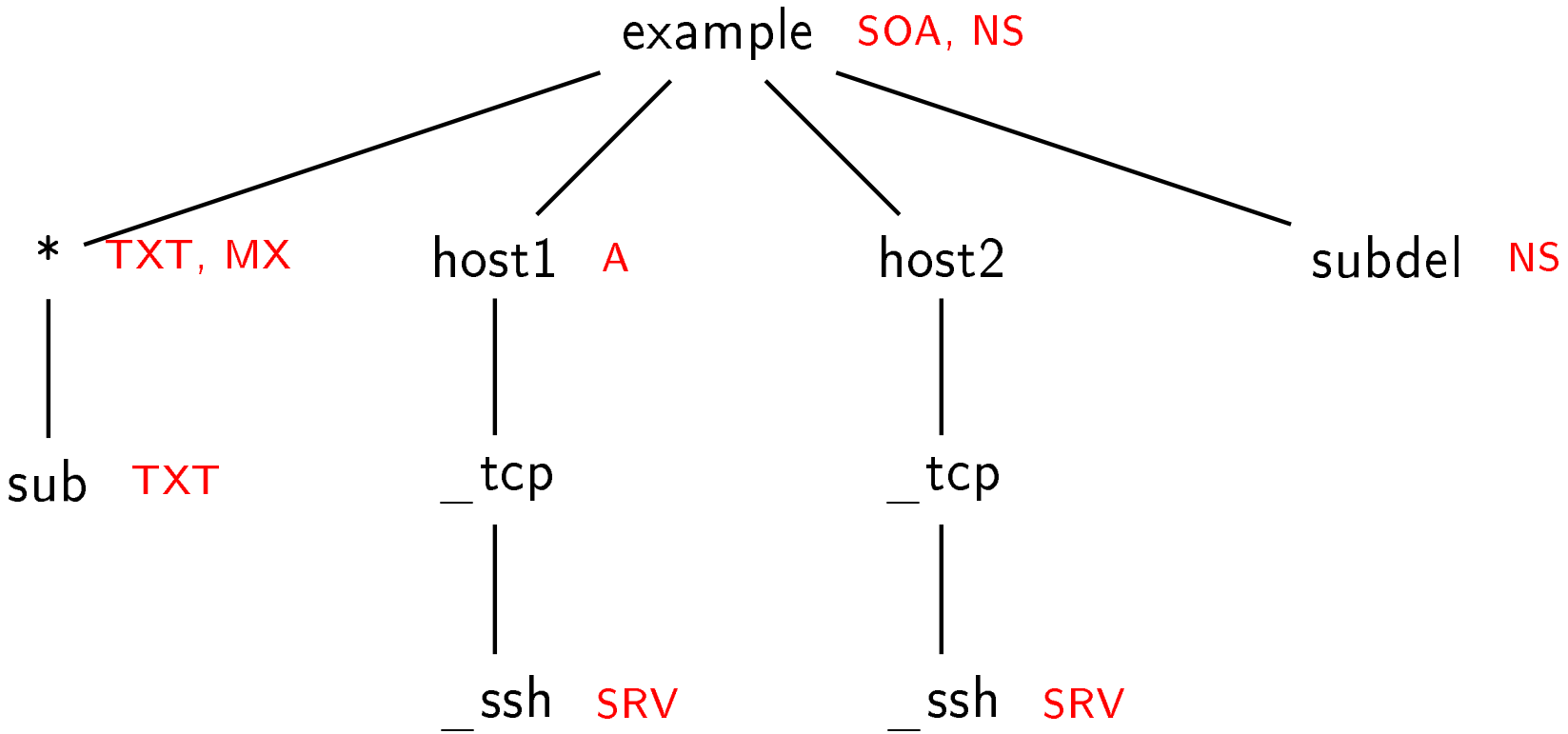
```
$ORIGIN example.
example.                    3600 IN  SOA   <SOA RDATA>
example.                    3600     NS    ns.example.com.
example.                    3600     NS    ns.example.net.
*.example.                  3600     TXT   "this is a wildcard"
*.example.                  3600     MX    10 host1.example.
sub.*.example.              3600     TXT   "... not a wildcard"
host1.example.              3600     A     192.0.2.1
_ssh._tcp.host1.example.    3600     SRV   <SRV RDATA>
_ssh._tcp.host2.example.    3600     SRV   <SRV RDATA>
subdel.example.             3600     NS    ns.example.com.
subdel.example.             3600     NS    ns.example.net.
```

RFC 4592 example tree



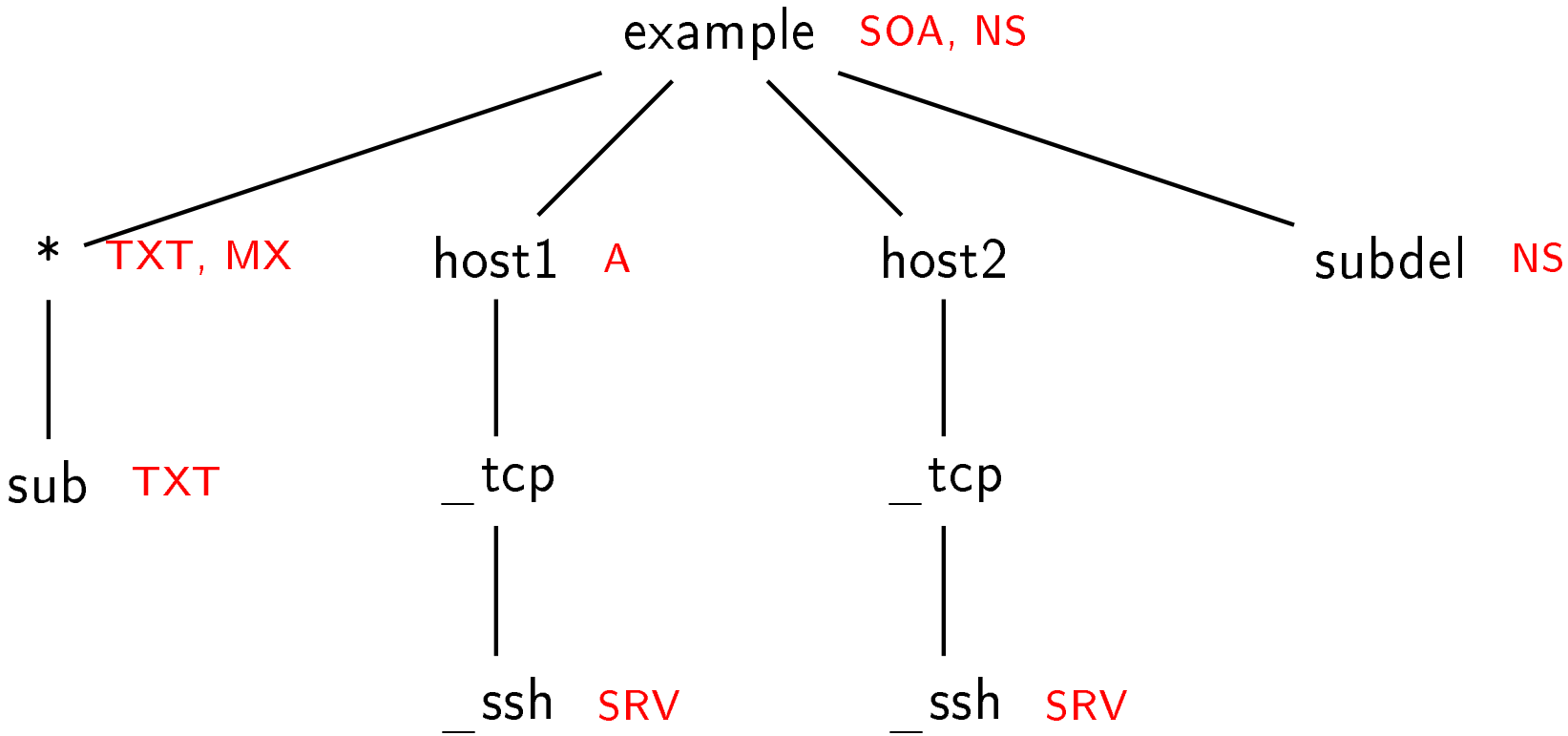
RFC 4592 example queries

QNAME	QTYPE	synthesized?	result
host3.example.	MX		



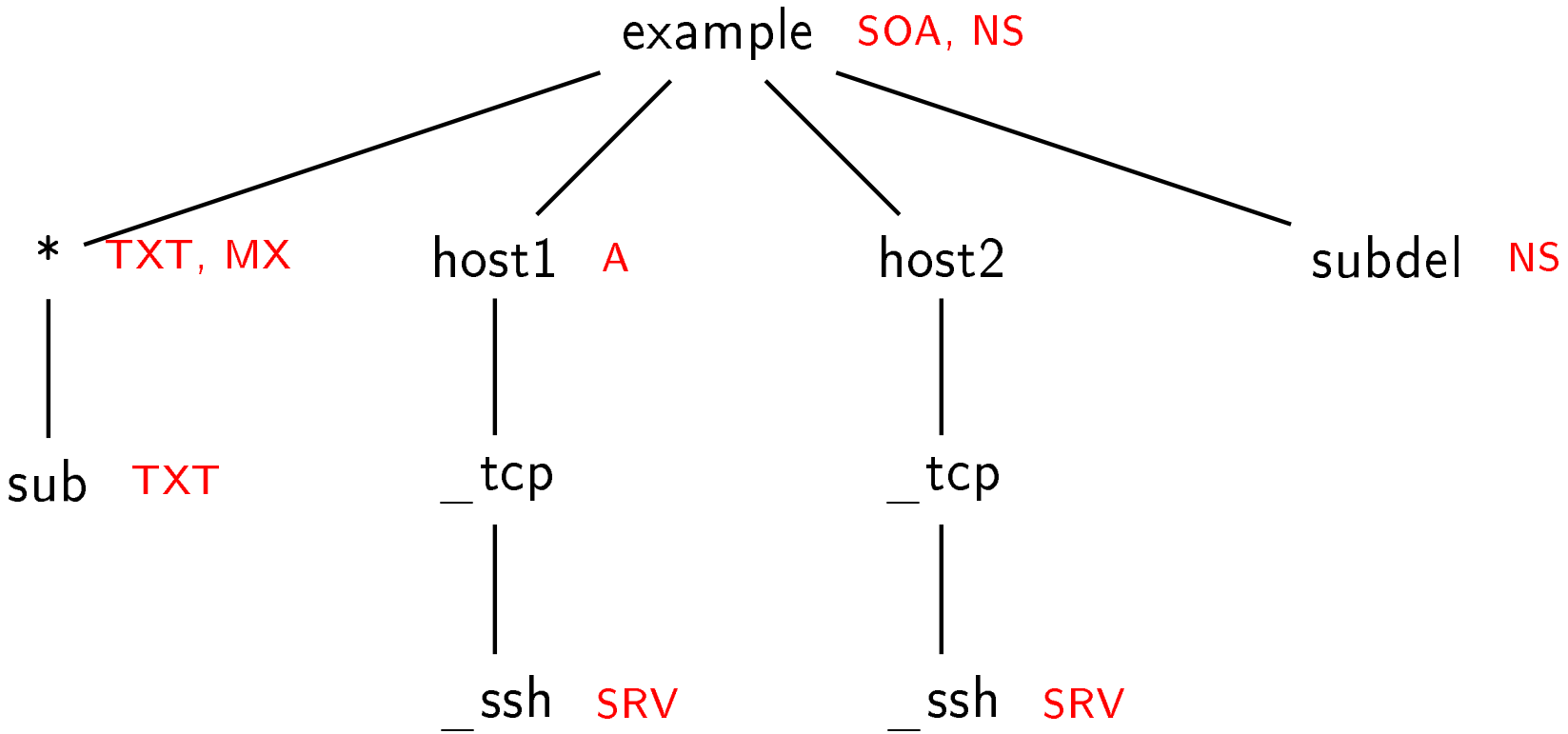
RFC 4592 example queries

QNAME	QTYPE	synthesized?	result
host3.example.	MX	yes	non-empty



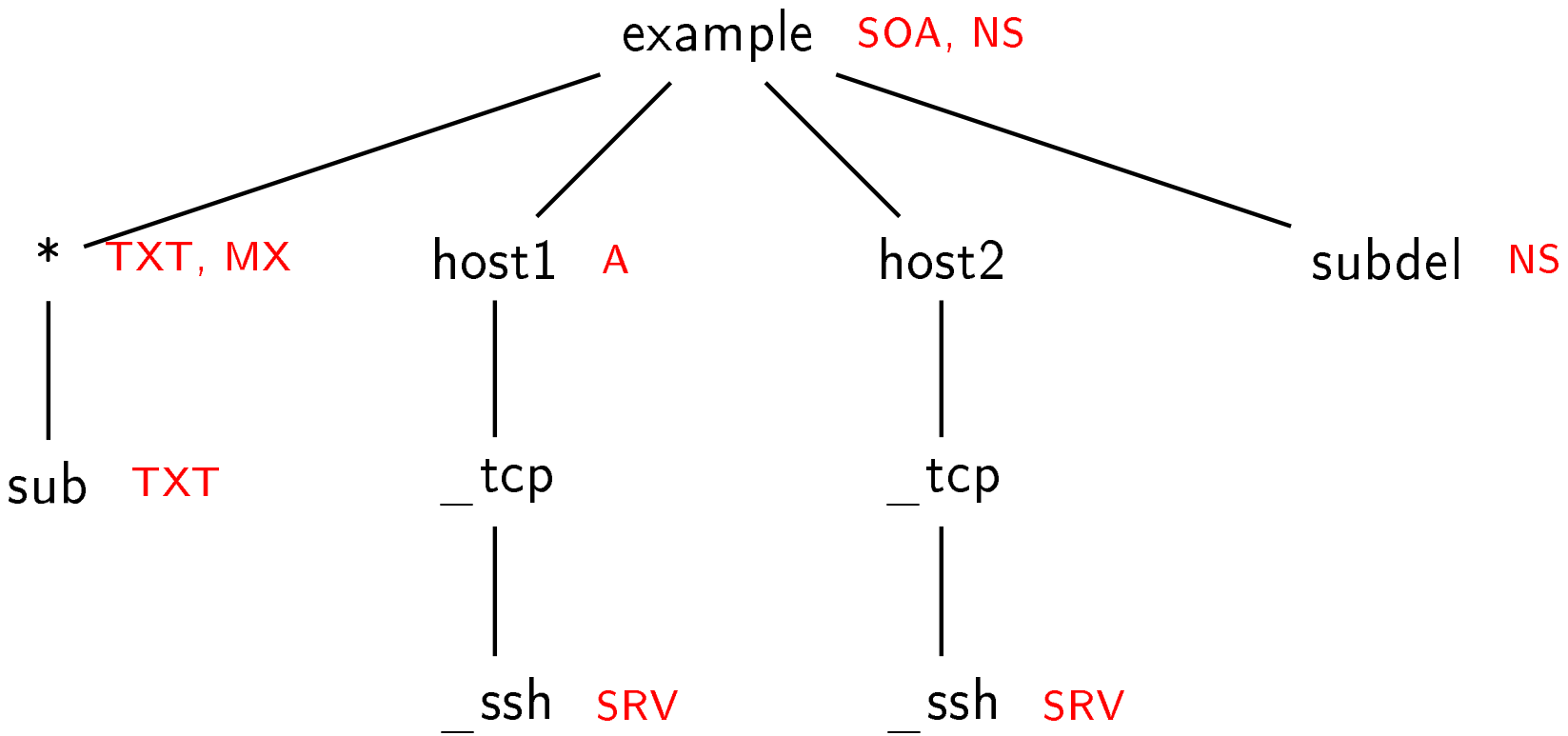
RFC 4592 example queries

QNAME	QTYPE	synthesized?	result
foo.bar.example.	TXT		



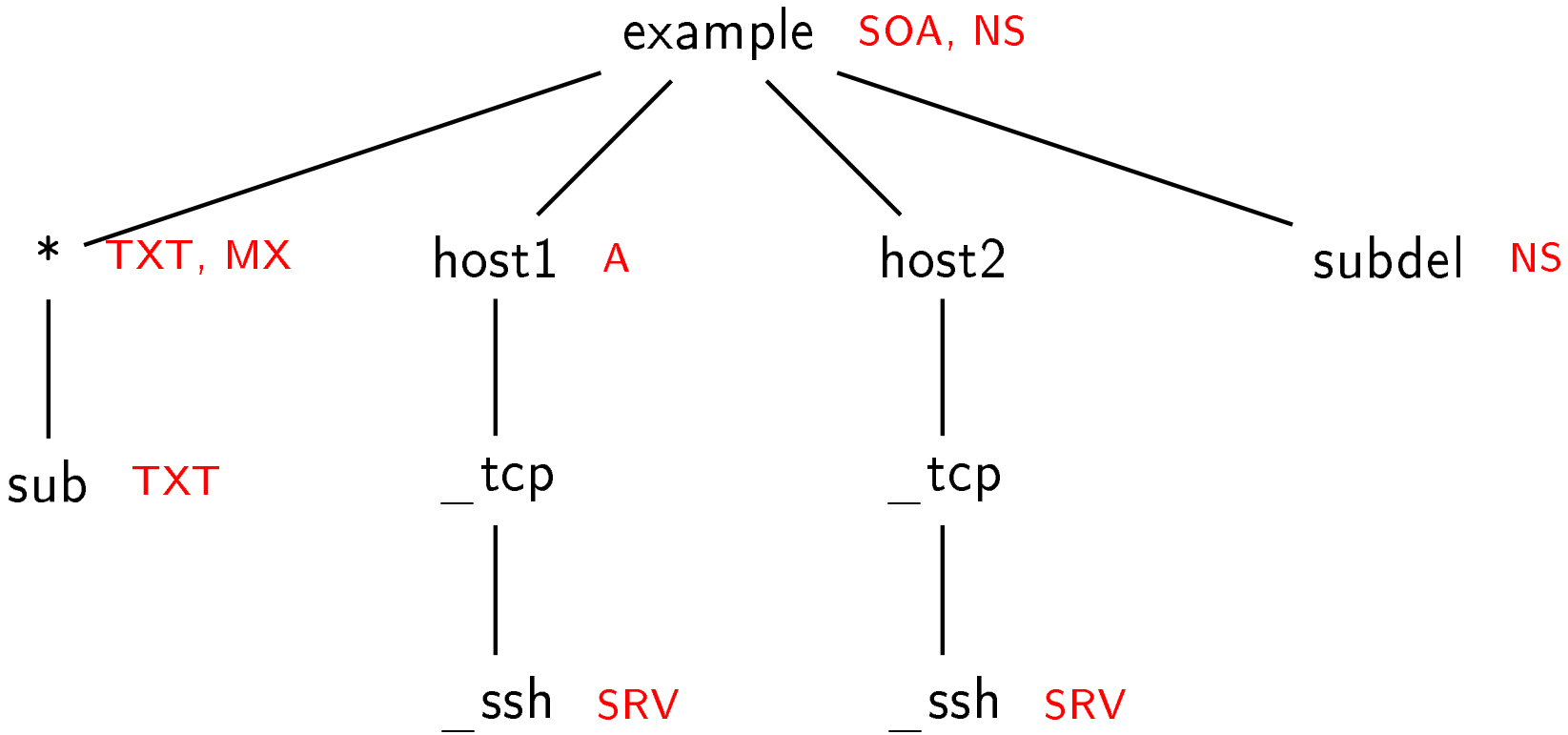
RFC 4592 example queries

QNAME	QTYPE	synthesized?	result
foo.bar.example.	TXT	yes	non-empty



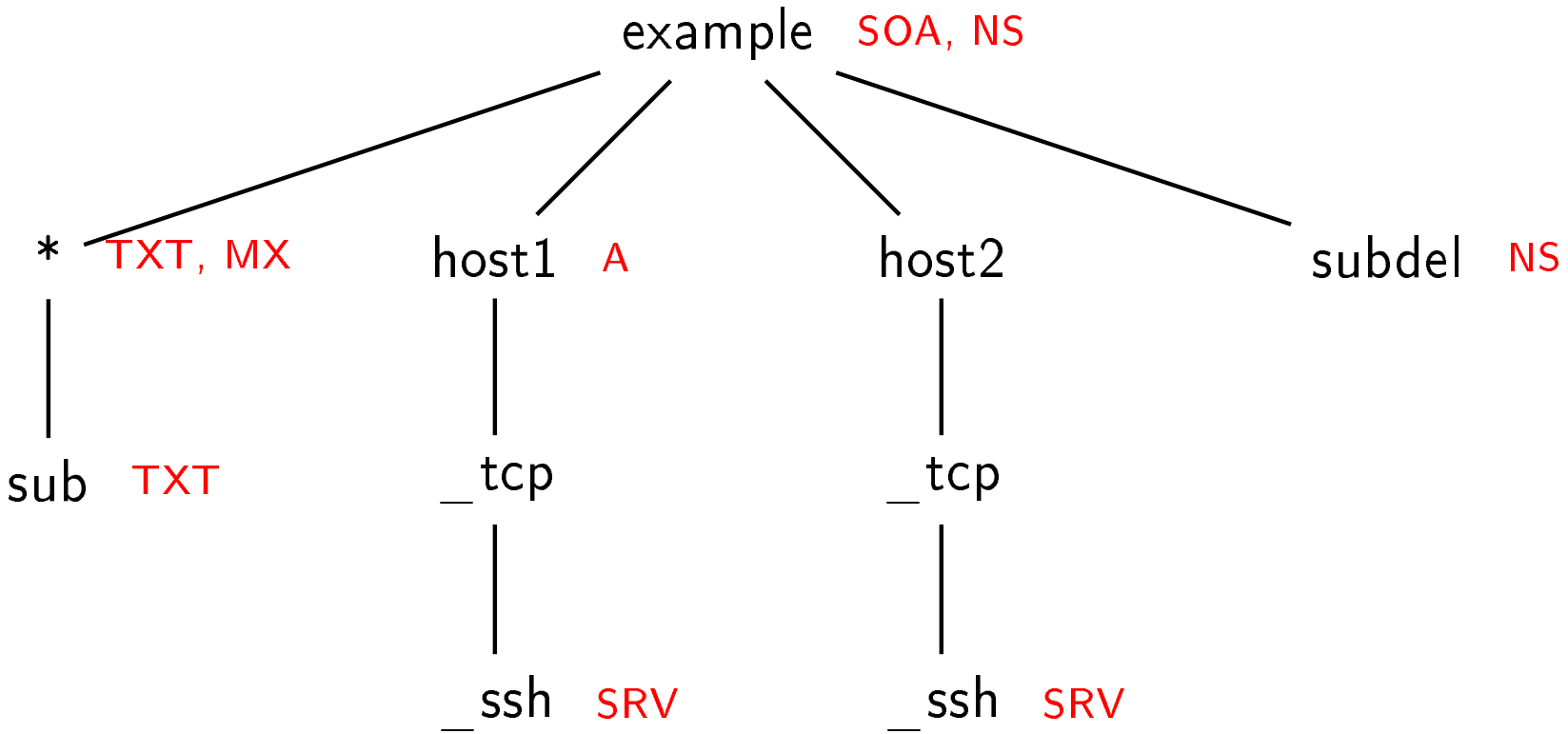
RFC 4592 example queries

QNAME	QTYPE	synthesized?	result
host1.example.	MX		



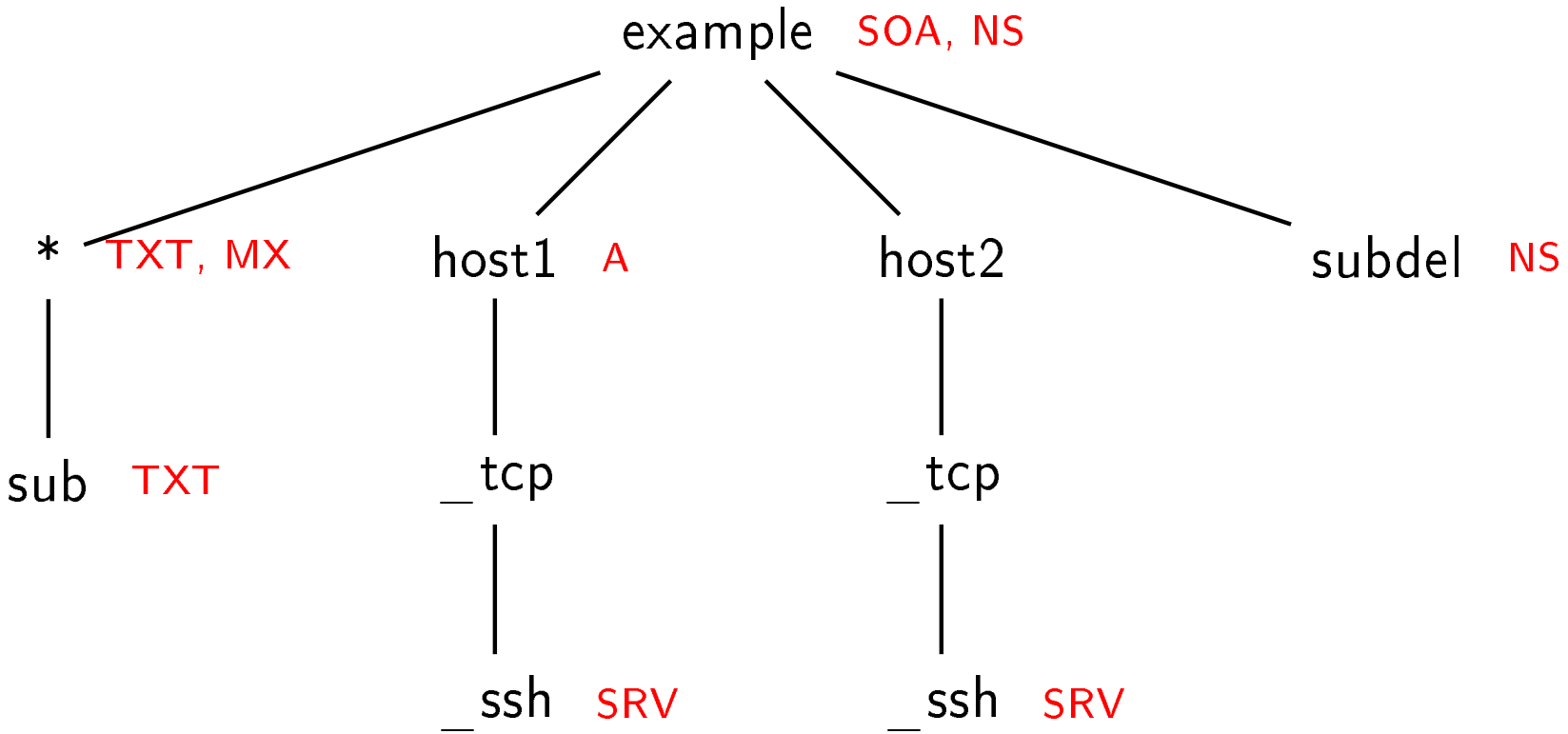
RFC 4592 example queries

QNAME	QTYPE	synthesized?	result
host1.example.	MX	no	empty



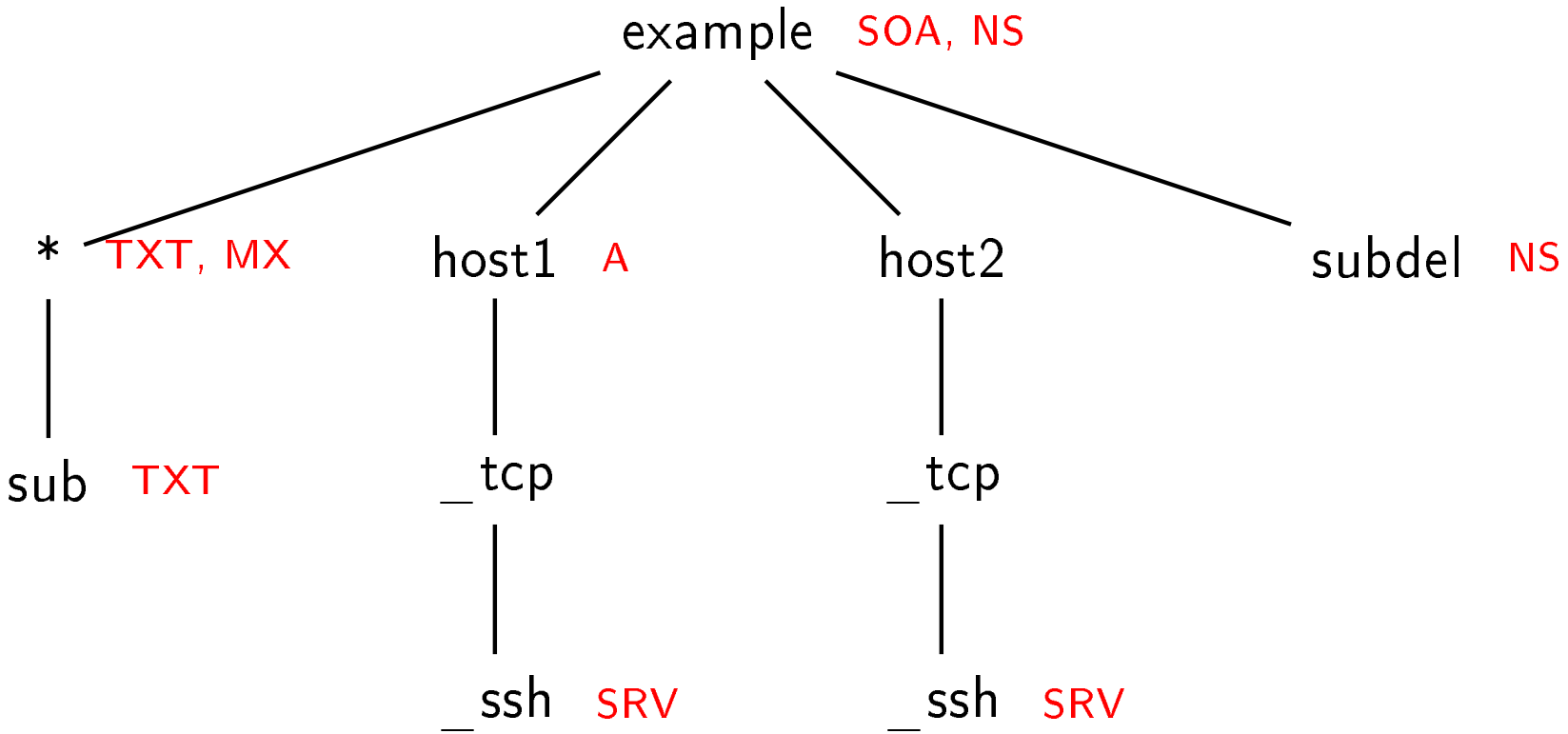
RFC 4592 example queries

QNAME	QTYPE	synthesized?	result
host3.example.	A		



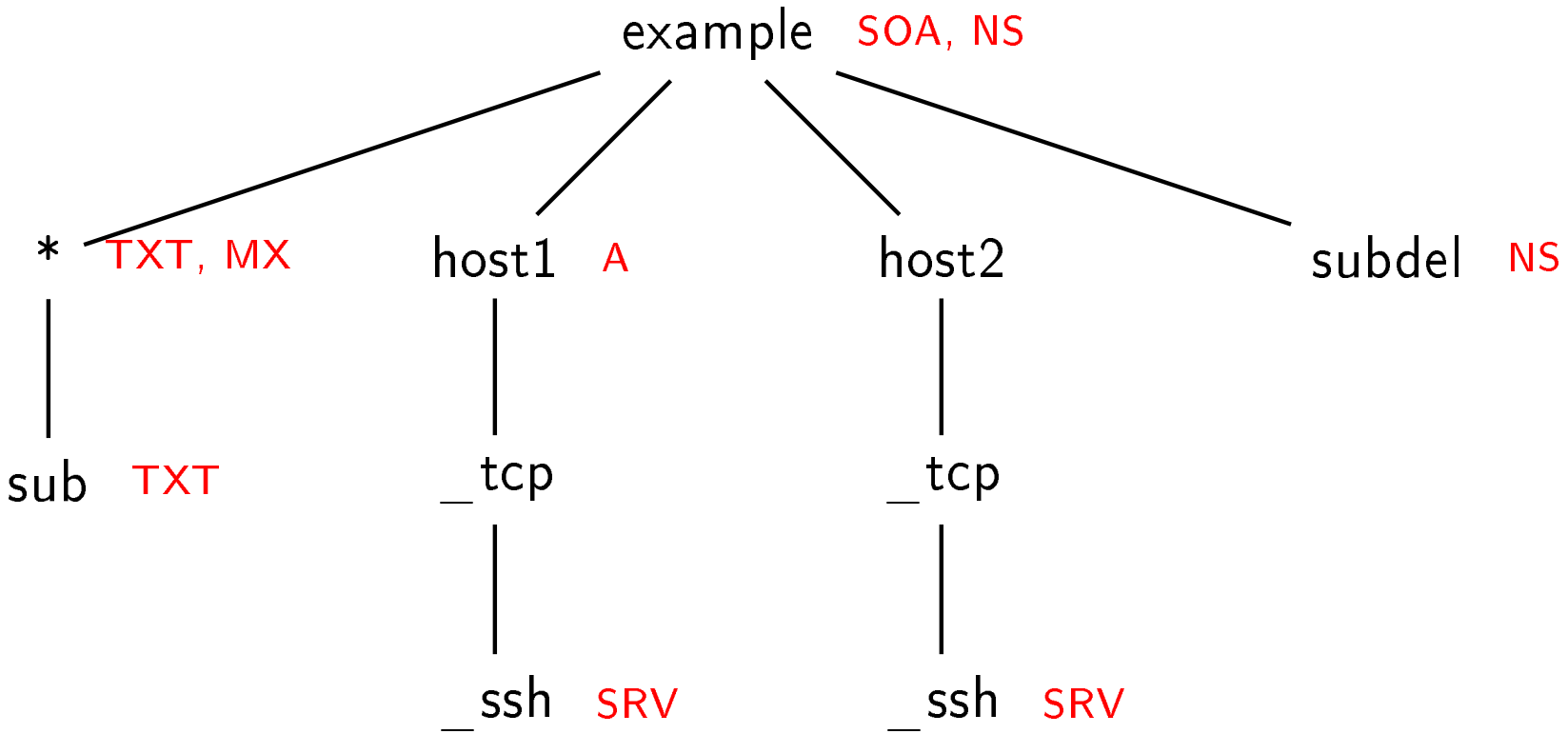
RFC 4592 example queries

QNAME	QTYPE	synthesized?	result
host3.example.	A	yes	empty



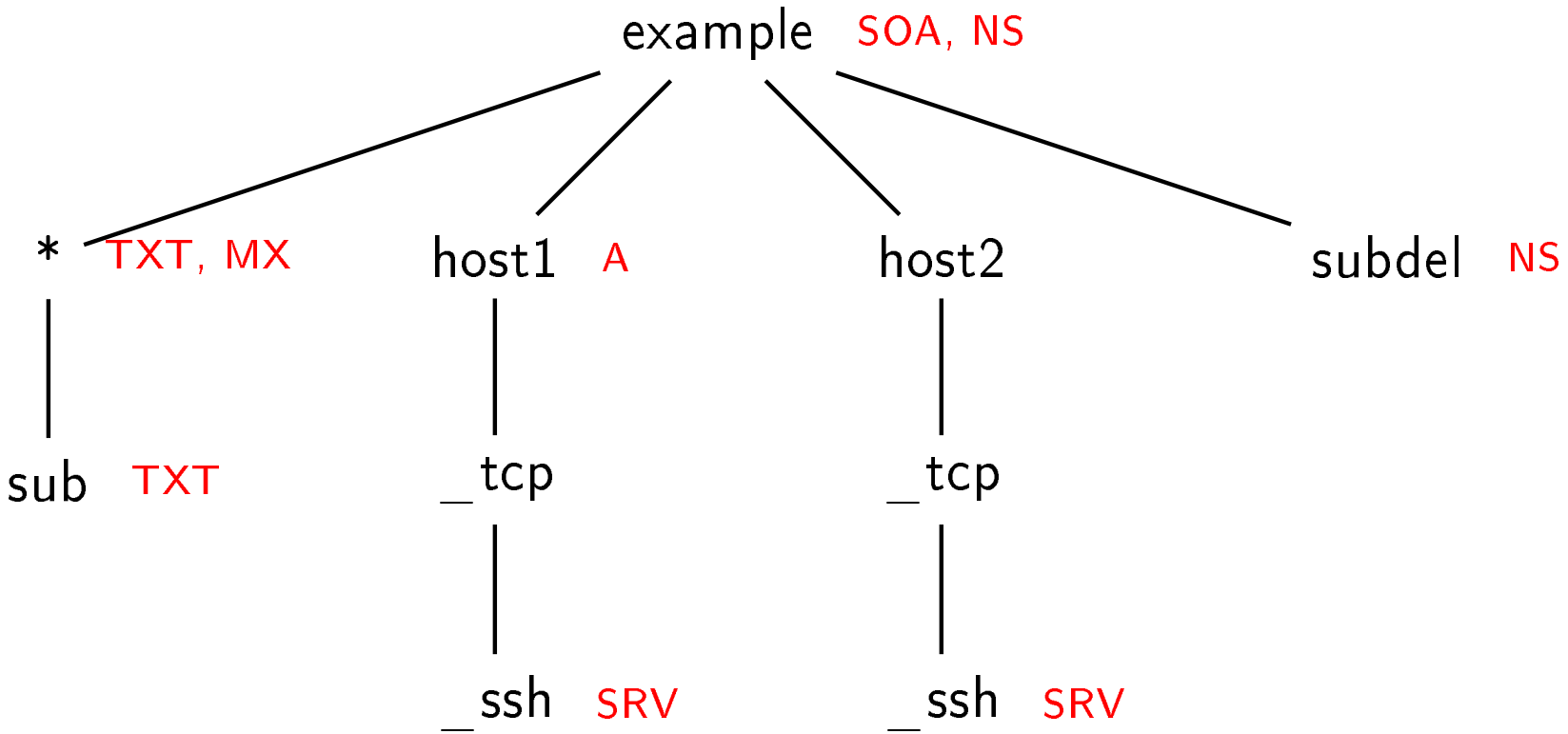
RFC 4592 example queries

QNAME	QTYPE	synthesized?	result
sub.*.example.	MX		



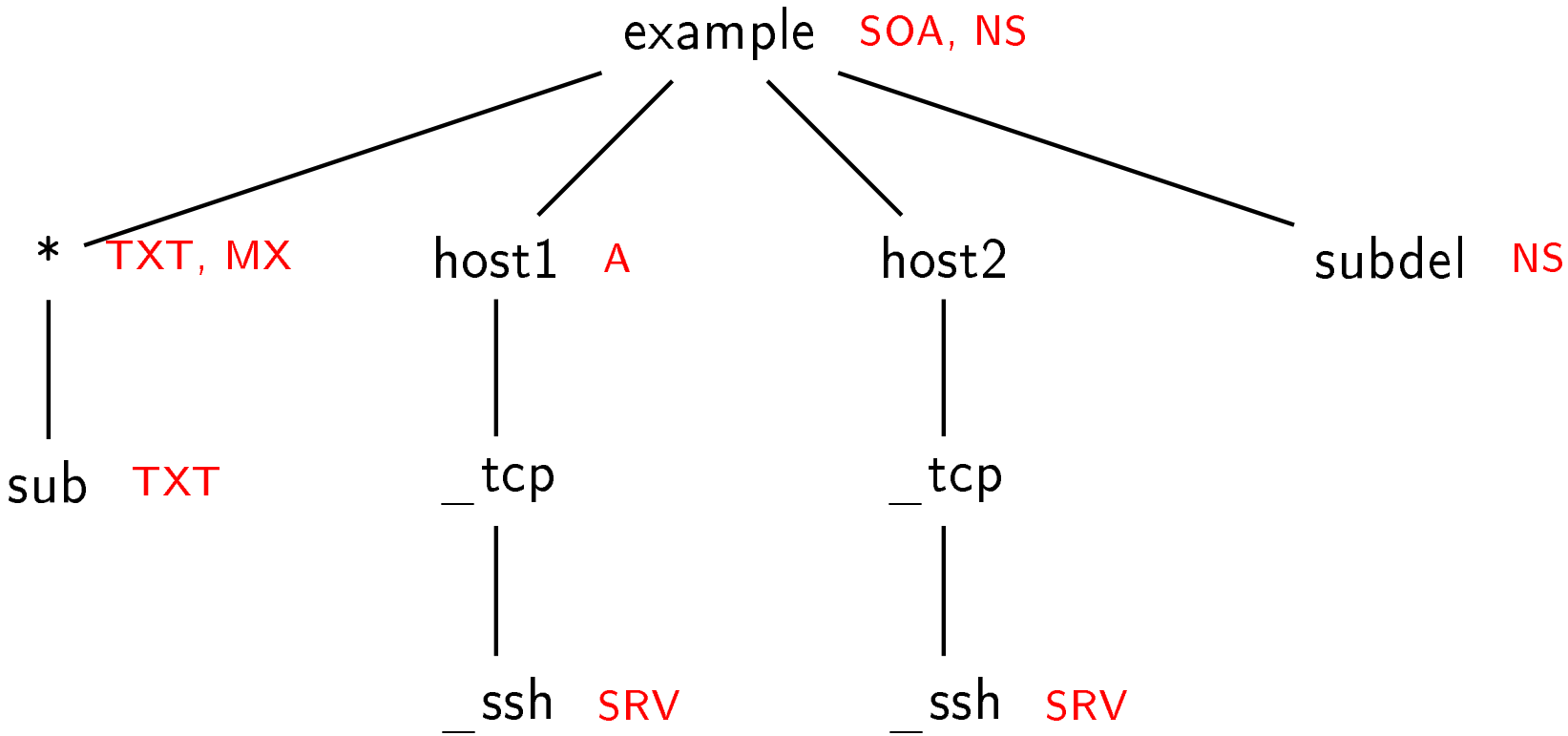
RFC 4592 example queries

QNAME	QTYPE	synthesized?	result
sub.*.example.	MX	no	empty



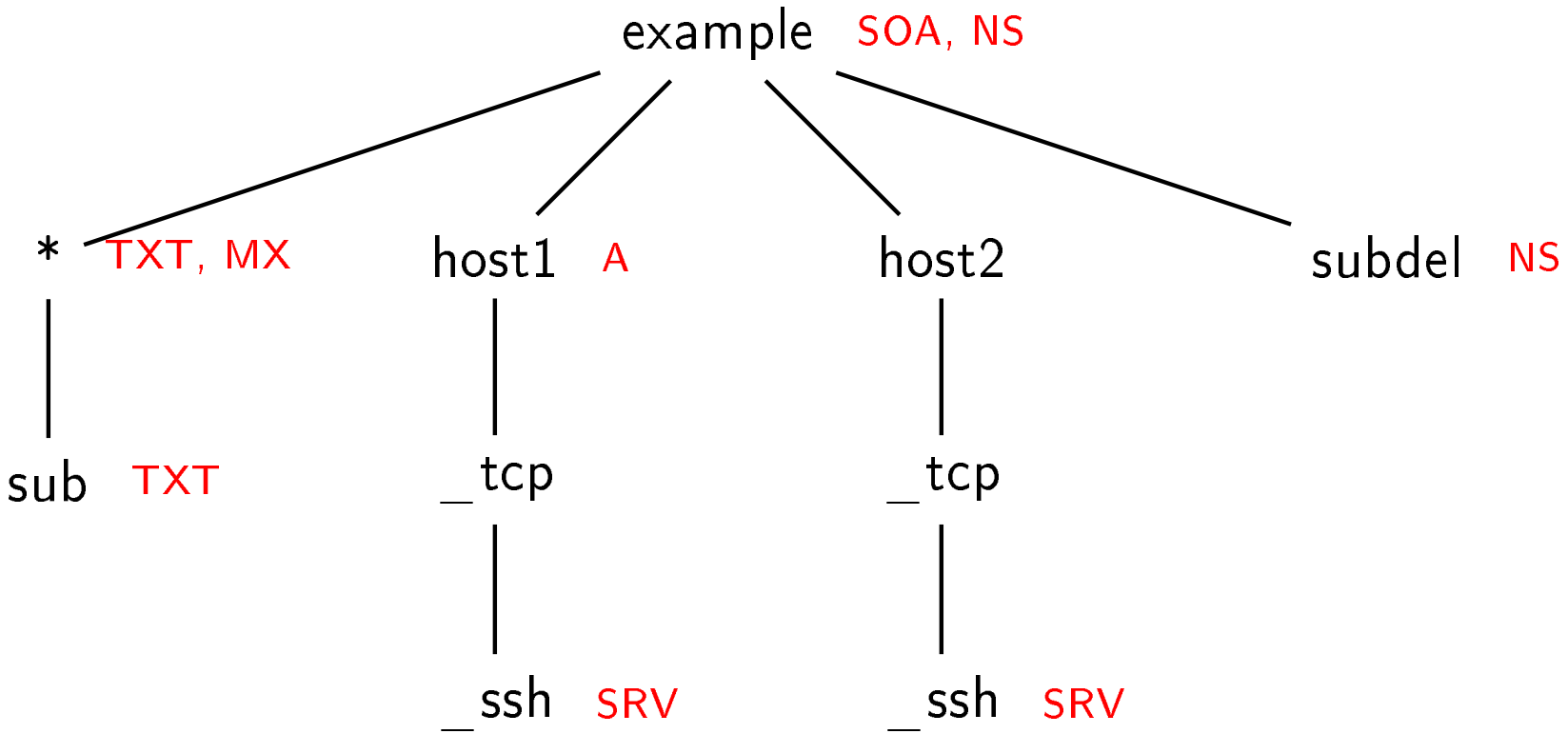
RFC 4592 example queries

QNAME	QTYPE	synthesized?	result
_telnet._tcp.host1.example.	SRV		



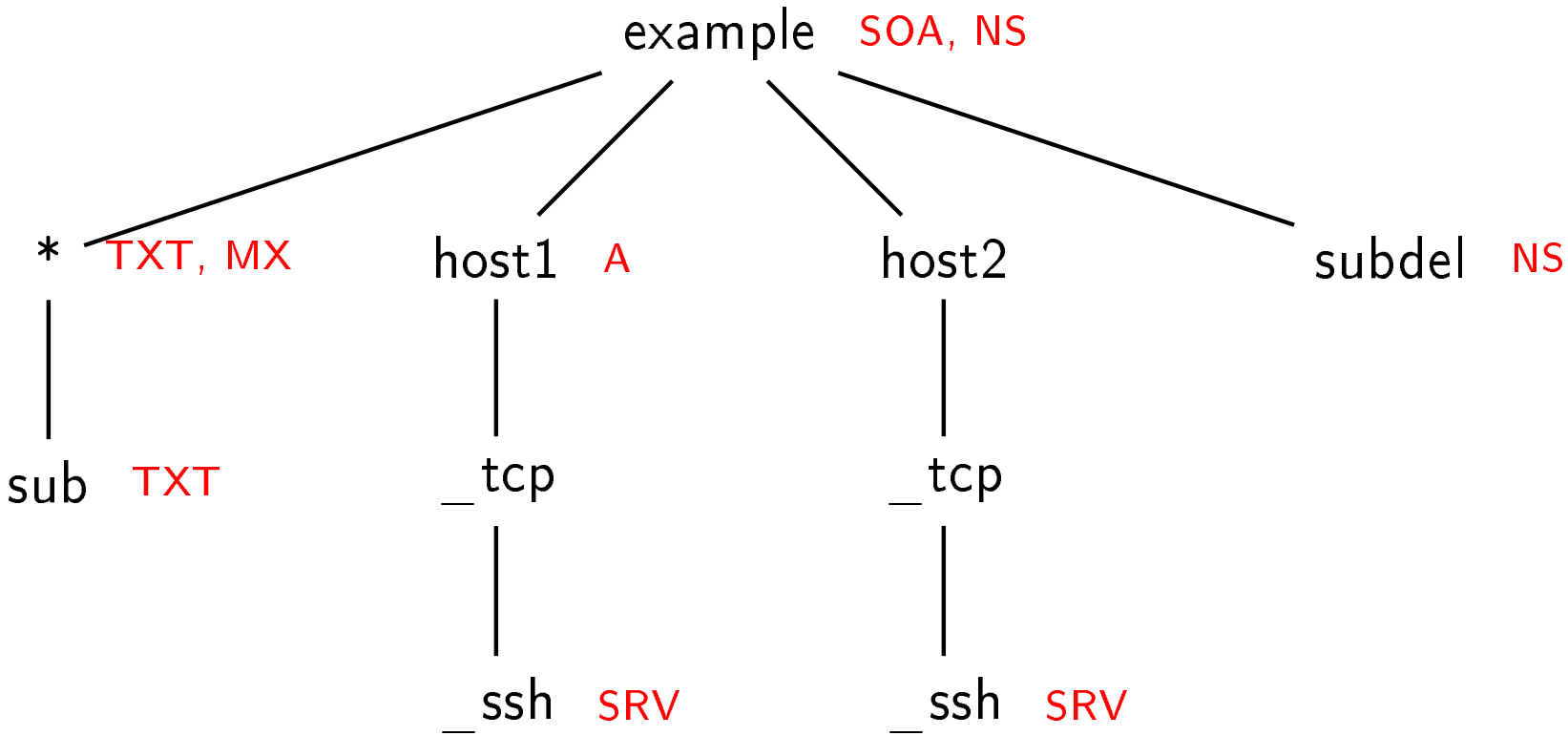
RFC 4592 example queries

QNAME	QTYPE	synthesized?	result
_telnet._tcp.host1.example.	SRV	no	no such domain



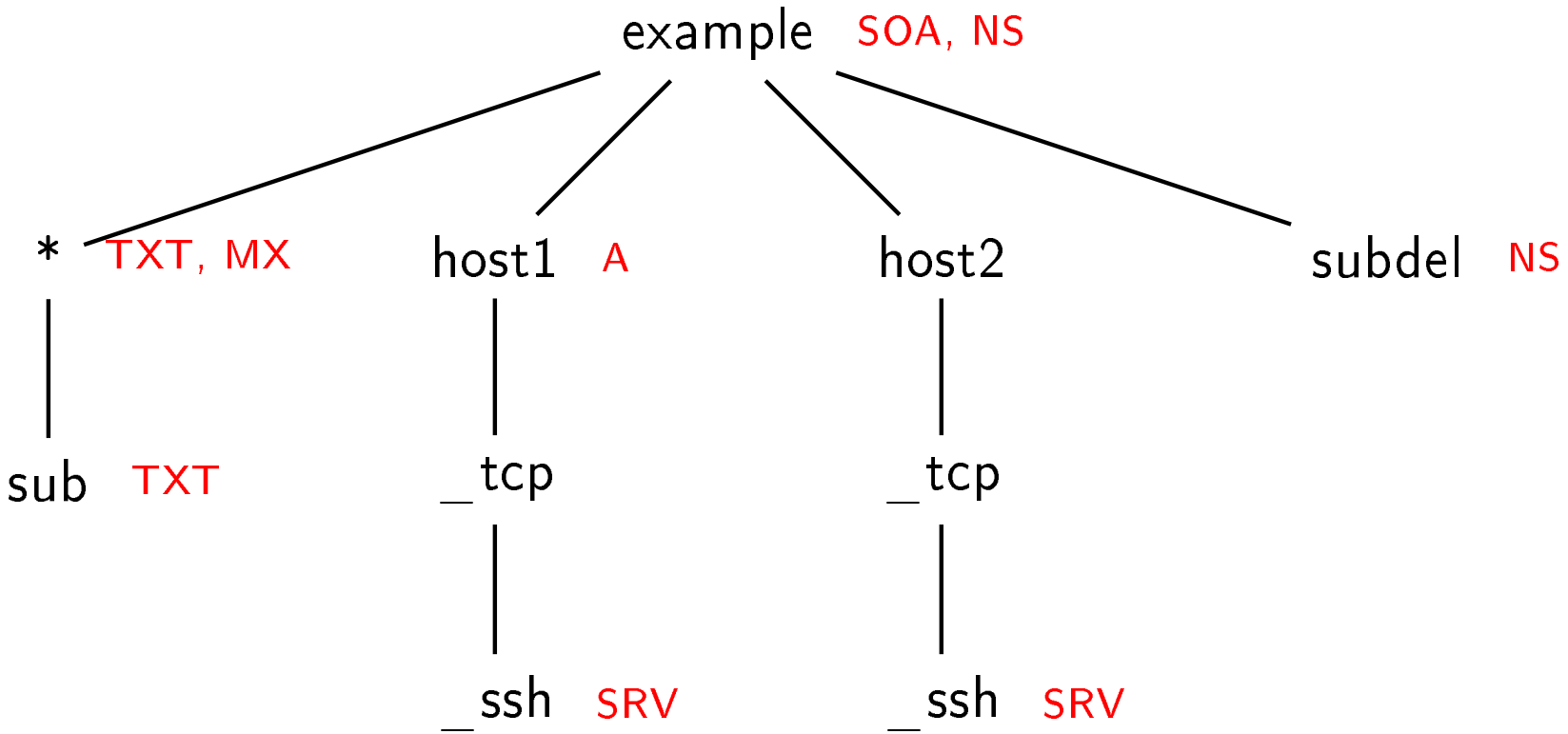
RFC 4592 example queries

QNAME	QTYPE	synthesized?	result
host.subdel.example.	A		



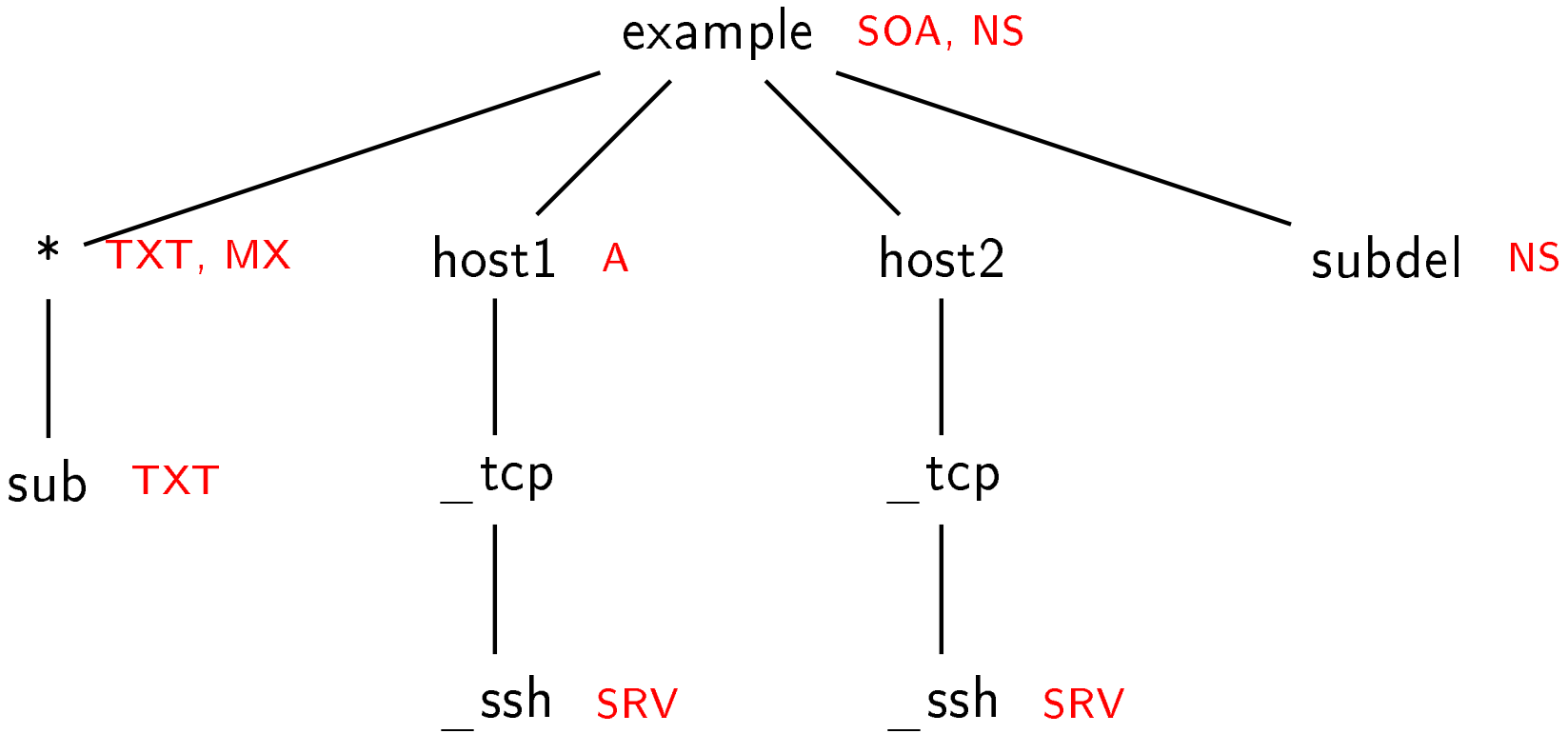
RFC 4592 example queries

QNAME	QTYPE	synthesized?	result
host.subdel.example.	A	no	referral



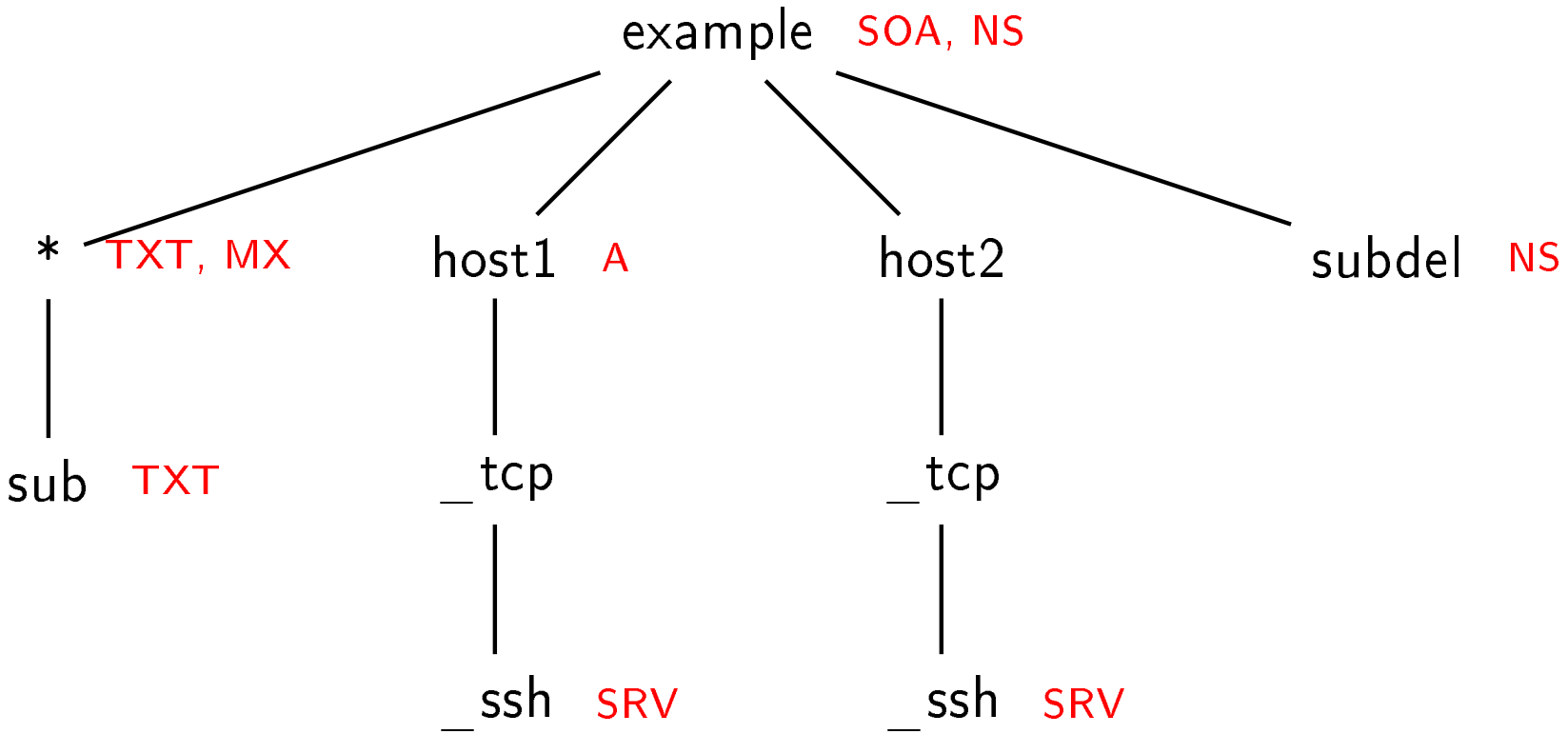
RFC 4592 example queries

QNAME	QTYPE	synthesized?	result
ghost.*.example.	MX		



RFC 4592 example queries

QNAME	QTYPE	synthesized?	result
ghost.*.example.	MX	no	no such domain



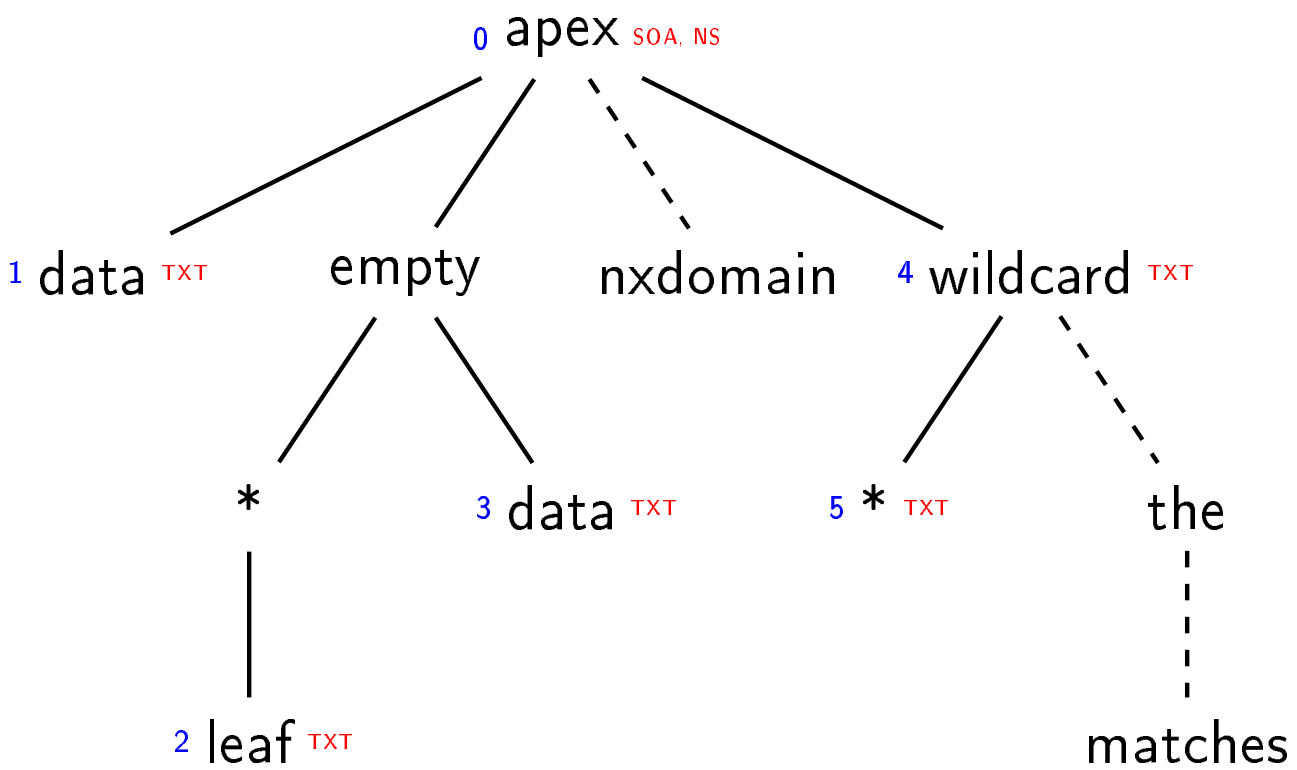
RFC 4592 example queries

QNAME	QTYPE	synthesized?	result
host3.example.	MX	yes	non-empty
host3.example.	A	yes	empty
foo.bar.example.	TXT	yes	non-empty
host1.example.	MX	no	empty
sub.*.example.	MX	no	empty
_telnet._tcp.host1.example.	SRV	no	no such domain
host.subdel.example.	A	no	referral
ghost.*.example.	MX	no	no such domain

Outline

Outline

Example zone



Proving "Nodata" with matching NSEC record

- "Nodata" means "No Error", but also no data
- So some QTYPE's have data, but the queried QTYPE has not
- The NSEC record at the owner contains a bitmap for existing types
  - Given the query "data.apex A"
  - Consider the NSEC record

    "data.apex NSEC leaf.*.empty.apex TXT"

Proving "Nodata" for empty non-terminals

- An empty non-terminal implies "No Error", but also no data
- But now no QTYPE's at all have data
- There is no NSEC record for the empty non-terminal
- The empty non-terminal is "covered" (skipped) by some NSEC record
  - Given the query "empty.apex A"
  - Consider the NSEC record

    "data.apex NSEC leaf.*.empty.apex TXT"
  - Server should include this NSEC record in the answer
  - Client should verify this NSEC record is the right one

Proving NXDOMAIN ("Name Error" or "No such domain")

- There is no NSEC record for the non-existing domain name
- The non-existing domain name is covered by some NSEC record
  - Given the query "nxdomain.apex A"
  - Consider the NSEC record

    "data.empty.apex NSEC wildcard.apex TXT"
  - Server should include this NSEC record in the answer
  - Client should verify this NSEC record is the right one
- What is the difference with the previous case?

Outline

NSEC records in the (positive) answer case

- If a domain name is synthesized from a wildcard record

  we also need proof that this has been done correctly
- The "next closer" is the last (non-existing) domain name

  on the path to the "closest encloser"
- The next closer domain name is covered by some NSEC record
  - Given the query "matches.the.wildcard.apex TXT"
  - Consider the NSEC record "*.wildcard.apex NSEC apex TXT"
  - Server should include this NSEC record in the answer
  - Client should verify this NSEC record is the right one

Proving "Nodata" for wildcard expansions

- This is a combination of previous cases
  - Use the NSEC type bitmap to prove queried QTYPE is not available
  - Use a covering NSEC RR for the next closer domain name
  - Given the query "matches.the.wildcard.apex A"
  - Consider the NSEC record "*.wildcard.apex NSEC apex TXT"
  - This sole NSEC record works for both requirements above
  - Server includes and client verifies this NSEC record
- What if the wildcard happens to be an empty non-terminal?

Proving NXDOMAIN revisited

- Find the NSEC record that covers the next closer domain name
- Moreover find the NSEC record that covers the (non-existing) source of synthesis
  - Given the query "nxdomain.apex A"
  - Consider the NSEC record

    "data.empty.apex NSEC wildcard.apex TXT"
  - And also the NSEC record

    "apex NSEC data.apex SOA NS"
  - Server includes and client verifies these NSEC records

NSEC3 records

- NSEC3 also creates a (circular) chain, but
  - uses hashes of the original domain names as labels

    for the owner domain names of the NSEC3 records
  - also hashes empty non-terminals
  - orders the hashes instead of the original domain names
  - uses the NSEC3PARAM type to specify hashing parameters

    and indicate the use of NSEC3 instead of NSEC
  - defines the Opt-Out flag to exclude insecure delegations from the chain