Information Network 1
Application protocols

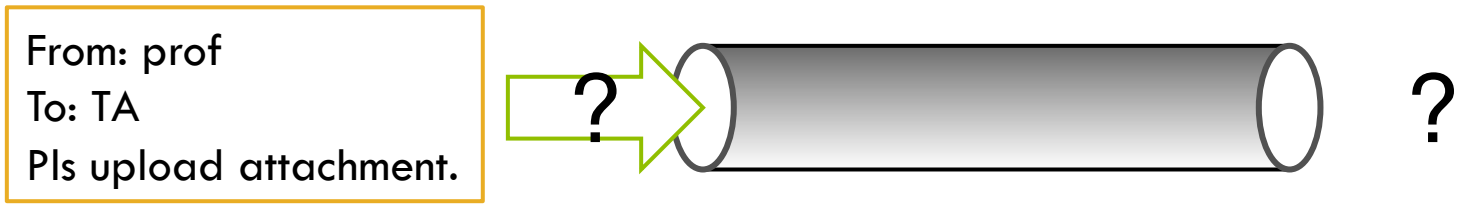
2012/5/23

Youki Kadobayashi

NAIST 奈良先端科学技術大学院大学

Application protocol challenges

- How do I send information?
  - Can I directly use TCP?
- How do I interact with the other end?
- How do I find the other end?



Encoding practices

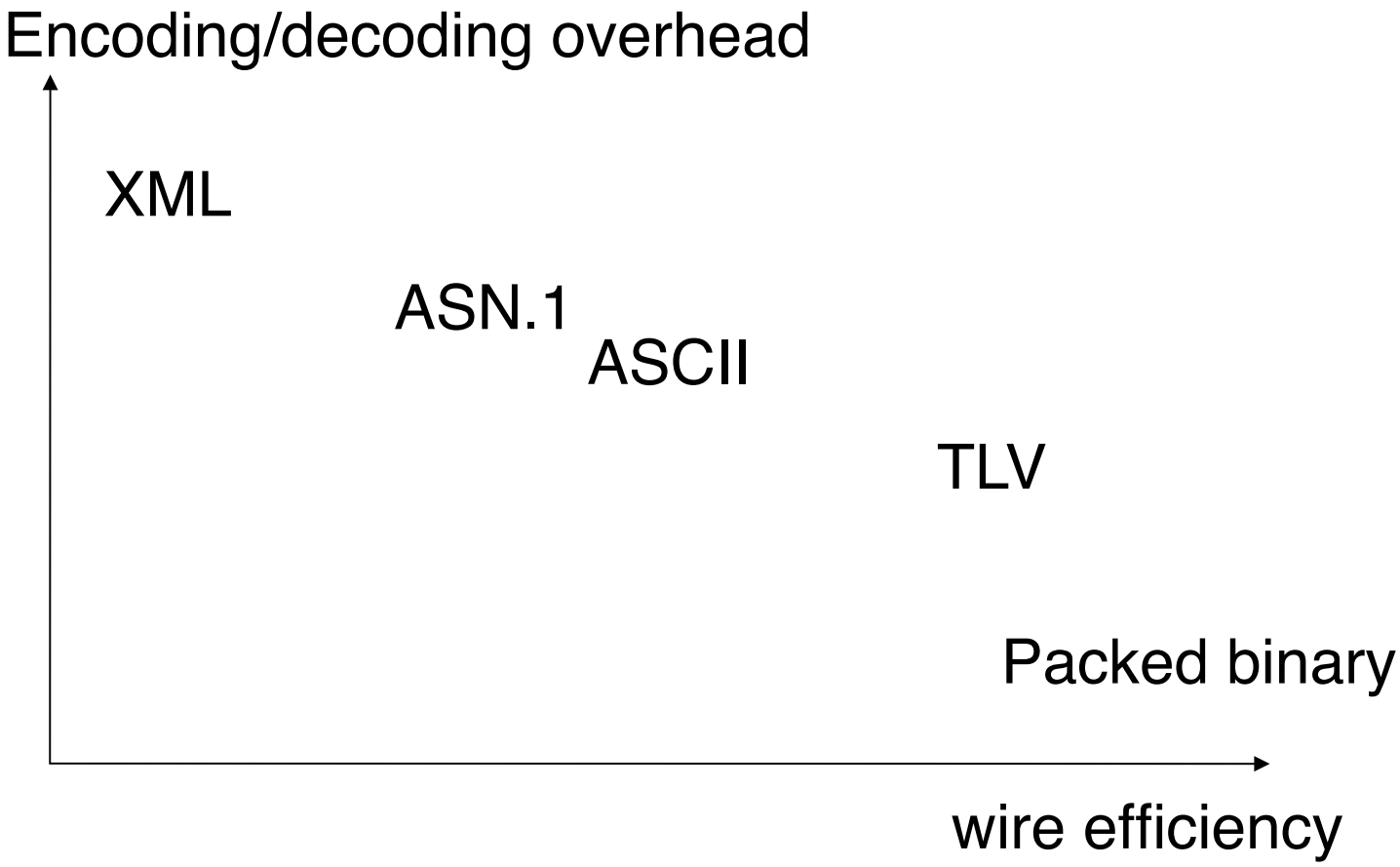
- Numeric 1 can be represented in many ways:
  - 00000001
  - 65 (ASCII code)
  - U+0041 (Unicode)
  - Type=1 Length=2 Value=1

→ Presentation layer

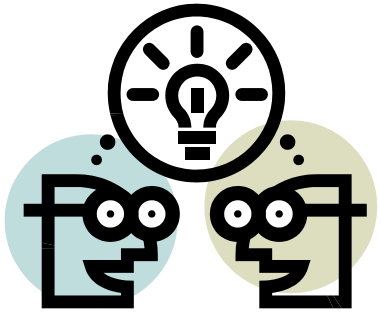
Established encoding practices in the presentation layer

- Binary
  - 00000001
- ASCII
  - 65
- TLV
  - Type=1 Length=2 Value=1
- ASN.1
  - → next lecture (network management)
- XML

Trade-offs in data representation



Q&A



 12/05/23

Encoding application protocols

Design choices

- Packed binary
- ASCII
- TLV
- XML
- ASN.1

… impacting:

- Efficiency
- Extensibility
- Readability
- Ease of definition
- Ease of parsing
- Ease of mixing
- Ease of validation

Packed binary

- Define bit boundaries
- Define semantics associated with each bit

- IP header (RFC791), TCP header (RFC793)
- Same technique can be employed in application layer, e.g., for sensor networks

16bit source port			16bit destination port
32bit sequence number			
32bit acknowledgment number			
4bit hlen	reserved	flags	16bit window size
16bit TCP checksum			16bit urgent pointer

20 octets

ASCII

- Consider byte-stream of ASCII characters as protocol
  - Point: make it human readable&writable
- ASCII protocols in the Internet
  - Mixture of machine-parseable status code and human-readable strings
- FTP (RFC959), SMTP (RFC2821) etc.
  - 3-digit status code, followed by greetings
  - 4-character command, followed by arguments

FTP: File Transfer Protocol

01.204.01754: 220 ftp.isi.edu NcFTPd Server (free educational license) ready.
76.020.00021: USER anonymous
01.204.01754: 331 Guest login ok, send your complete e-mail address as password.
76.020.00021: PASS -wget@
01.204.01754: 230 Logged in anonymously.

76.020.00021: SYST
01.204.01754: 215 UNIX Type: L8
76.020.00021: PWD
01.204.01754: 257 "/" is cwd.
76.020.00021: TYPE I
01.204.01754: 200 Type okay.
76.020.00021: CWD /in-notes
01.204.01754: 250 "/in-notes" is new cwd.

SMTP: Simple Mail Transfer Protocol


```
1758: 220 ns.iij-mc.com ESMTP Sendmail 8.9.3p2+3.1W/3.7W/ns; Fri, 30 May 2003 00:15:43 +09
0025: EHLO mf.aist-nara.ac.jp
1758: 250 ns.iij-mc.com Hello 168.pool3.ftthtokyo.att.ne.jp [165.76.67.168], pleased to meet you
0025: MAIL FROM:<youki@is.aist-nara.ac.jp> SIZE=1524
1758: 250 <youki@is.aist-nara.ac.jp>... Sender ok
0025: RCPT TO:<bunji@iij-mc.com>
1758: 250 <bunji@iij-mc.com>... Recipient ok
0025: DATA
1758: 354 Enter mail, end with "." on a line by itself
0025: Received: from mf.aist-nara.ac.jp (localhost [127.0.0.1])

3 +0900 (JST)
```

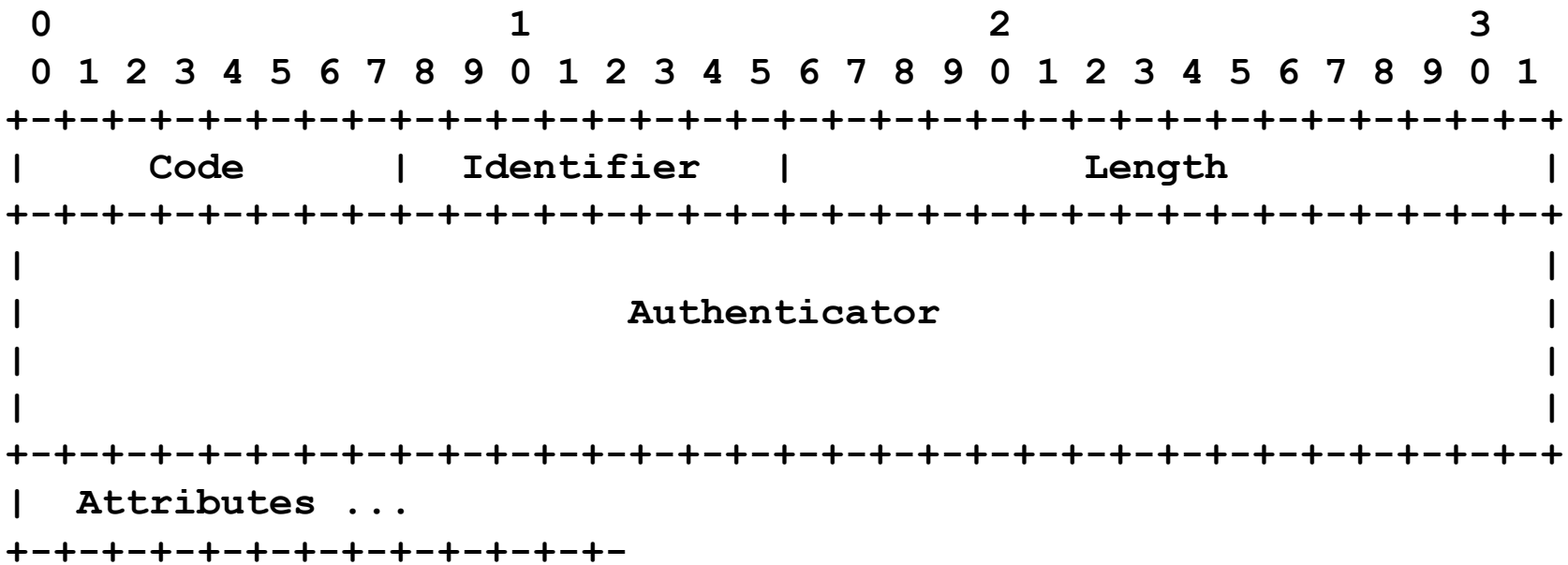

TLV

- Type, Length, Value
  - Pros:
    Compact, yet extensible
  - Cons:
    All data types have to be defined in protocol spec

- RADIUS (RFC2138), OSPF (RFC2328) etc.
  - Bit representation of TLV

RADIUS Packet Format

- (RFC2138) 3. Packet Format



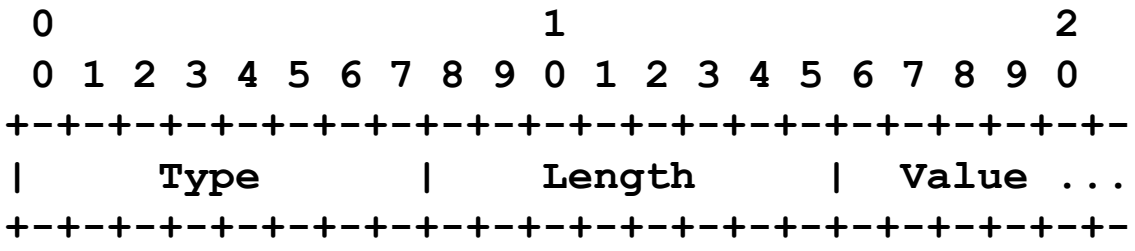
Code:

1 Access-Request

2 Access-Accept

3 Access-Reject

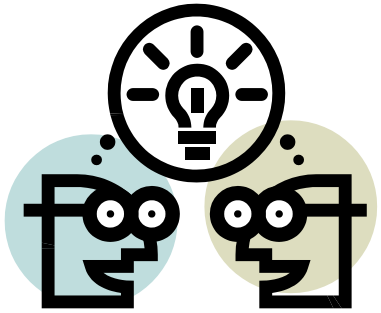
RADIUS Attribute TLV



Type

1	User-Name
2	User-Password
3	CHAP-Password
4	NAS-IP-Address

Q&A

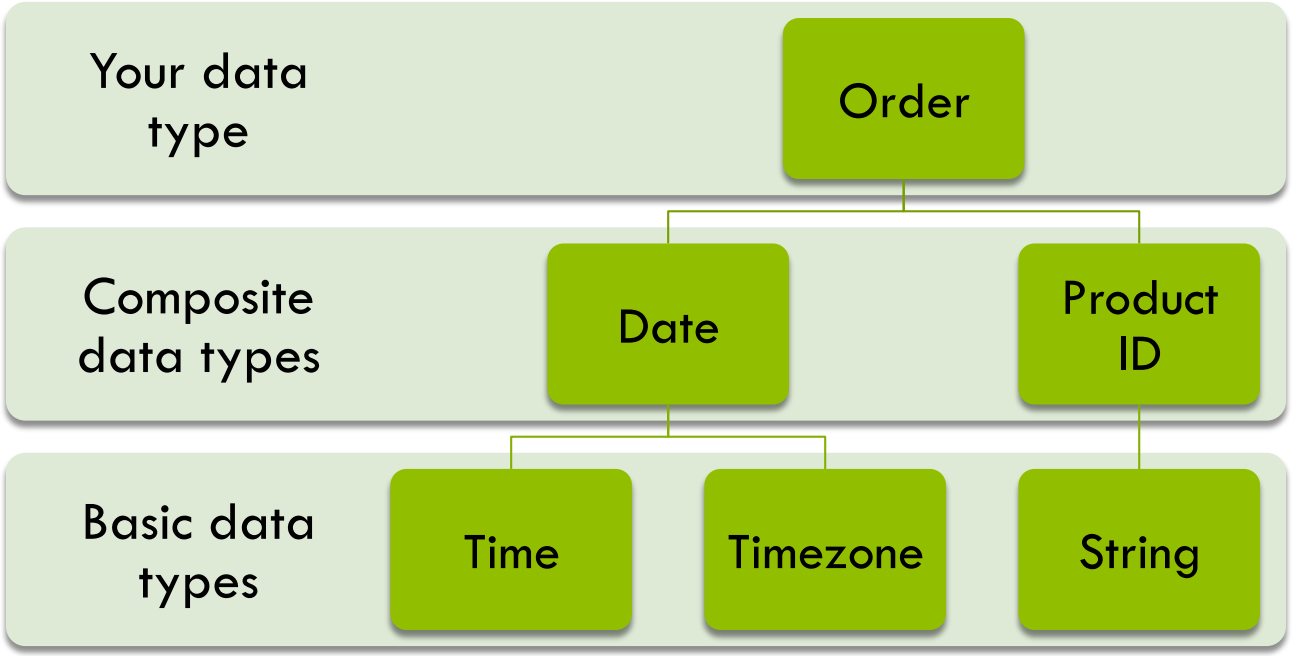


 12/05/23

XML: Extensible data types

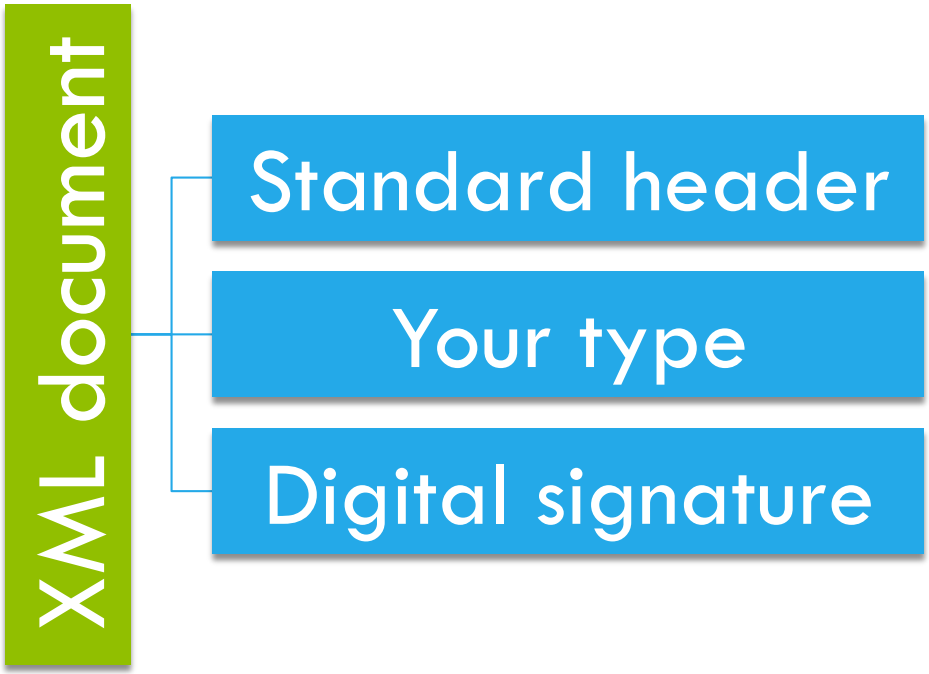
- Data types (static / dynamic typing)
- Definition of namespace
- Data that describes itself
- Rich set of tools and libraries

XML: Reuse of data types



- You don't need to redefine "Date" type in order to define your own type

XML: Modular data types



- You can reuse existing standard for document structures, digital signature, etc.

XML message format

- <?xml version = "1.0"?>
- <tag attribute="value">
  <another-tag another-attribute="value" />
- </tag attribute="value">

Management of namespace through XML Namespace


```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:xsi='http://www.w3.org/1999/XMLSchema-instance'
xmlns:xsd='http://www.w3.org/1999/XMLSchema' xmlns:soapenc='http://
schemas.xmlsoap.org/soap/encoding/' soap:encodingStyle='http://
schemas.xmlsoap.org/soap/encoding/'>
 <soap:Body>
  <n:getQuoteResponse xmlns:n='urn:xmethods-delayed-quotes'>
  <Result xsi:type='xsd:float'>7.92</Result>
  </n:getQuoteResponse>
 </soap:Body>
</soap:Envelope>
```


Describing data types in XML: XML Schema


```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/1999/
XMLSchema-instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/
soap/encoding/" xmlns:xsd="http://www.w3.org/1999/XMLSchema“
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Body>
  <namesp1:getQuote xmlns:namesp1="urn:xmethods-delayed-quotes">
  <symbol xsi:type="xsd:string">AKAM</symbol>
  </namesp1:getQuote>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


XML: other features

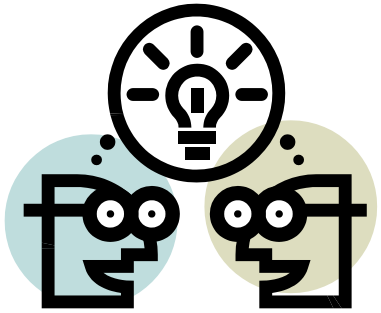
- Self-descriptive: XML Schema
- Transformation tools: XPath, XSLT, etc.
- Rich set of API: DOM, SAX, etc.

- Message integrity and confidentiality:
  - XML Digital Signature
  - XML Encryption

Presentation layer summary: a birds-eye view

Approach	Fixed	Extensible	
	Statically typed		Dynamically typed
binary	Packed binary	TLV	ASN.1
string		ASCII	XML

Q&A



 12/05/23

How do I interact with the other end?

- Who will be initiator?
- Who will be responder?

- What kind of information will be sent?
- Any state transitions?

- Essentially you are designing protocol FSM.

- Protocol FSM may deadlock, without careful design.

Ensuring desirable protocol properties

- Liveness property
  - Something good eventually happens
  - e.g., completion of data transmission
- Safety property
  - Something bad never happens
  - e.g., starvation
- → model checker

Ensuring desirable protocol properties with model checker


```
chan q[2] = [MaxSeq] of { byte, byte };   /* message passing channel */

active [2] proctype p5()     /* starts two copies of proctype p5 */
{ byte NextFrame, AckExp, FrameExp, r, s, nbuf, i;
  chan in, out;
  in = q[_pid];
  out = q[1-_pid];
  xr in; xs out;          /* partial order reduction claims */

  do
  :: nbuf < MaxSeq ->    /* outgoing messages */
       nbuf++;
       out!NextFrame,    (FrameExp + MaxSeq) percent (MaxSeq + 1);
       inc(NextFrame)

  :: q[_pid]?r,s ->        /* incoming messages */
       if
       :: r == FrameExp ->
            printf("MSC: accept percentd\n," r);
            inc(FrameExp)
       :: else /* ignore message */
       fi;
       do
       :: ((AckExp <= s) && (s < NextFrame))
       || ((AckExp <= s) && (NextFrame < AckExp))
       || ((s < NextFrame) && (NextFrame < AckExp)) ->
             nbuf--;
             inc(AckExp)
       :: else -> break
       od
```

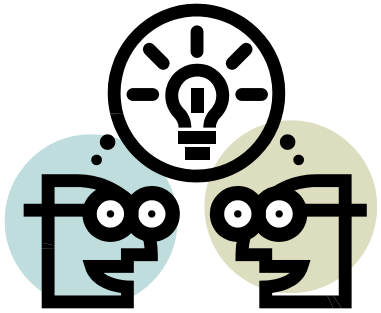
Appendix C: Verification model of a sliding window protocol

For further study:

- **LTL**
- **Model checking**

Source: The model checker SPIN, IEEE Trans. on Software Engineering, May 1997

Q&A



 12/05/23

How do I find the other end?

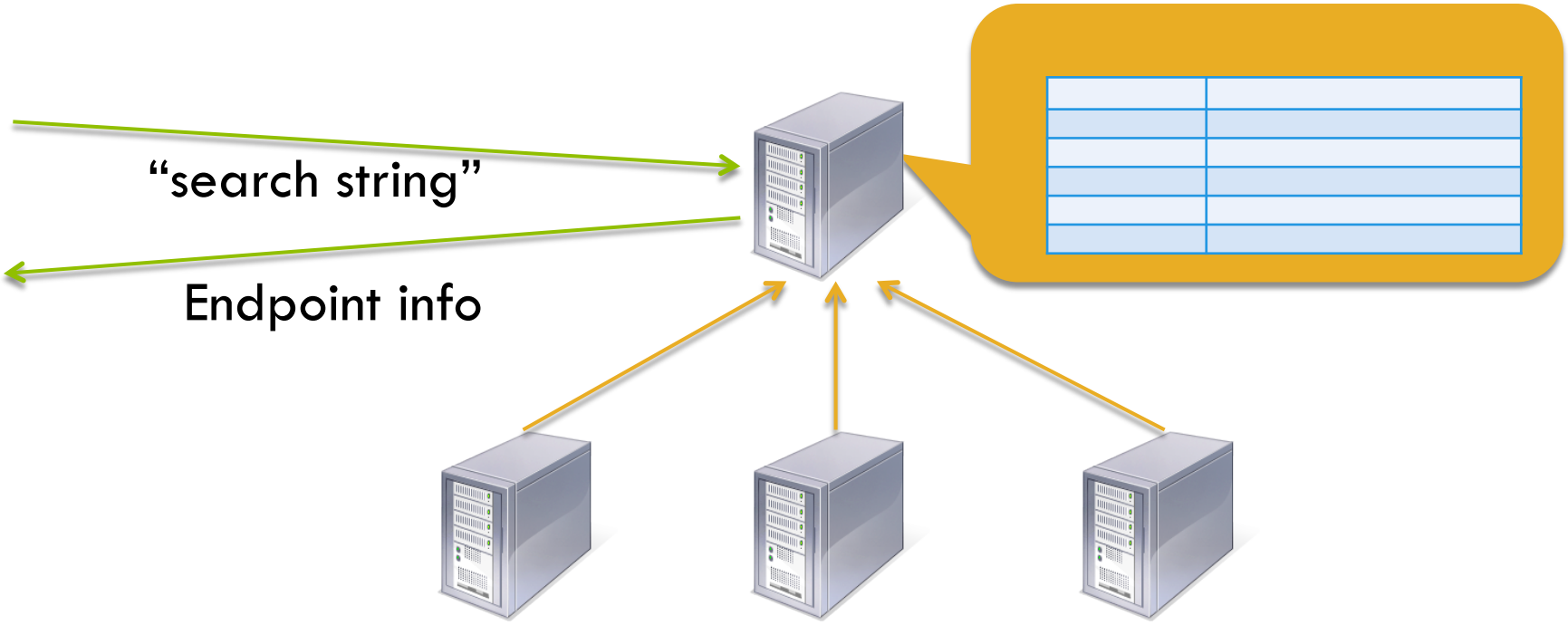
- 1. the other end is known *a priori*.
- 2. central authority knows the other end.
- 3. the other end has to be somebody who processes this information.

A priori knowledge of endpoint

- e.g., by Uniform Resource Locator (RFC1738)
  - Type of application protocol
  - TCP port number
  - Host name
  - etc.

```
      http://example.com:8080/over/there?name=ferret#nose
       \_/   \______________/\_________/ \_________/ \__/
        |           |            |            |        |
     scheme     authority       path        query   fragment
```

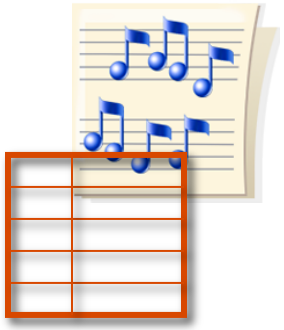

A central authority for endpoint lookup



- Many algorithms for scaling beyond 1000 servers
- For further study: NSDI, SIGCOMM papers

Discovery of endpoints from information types

- Discovery of services associated with the resource

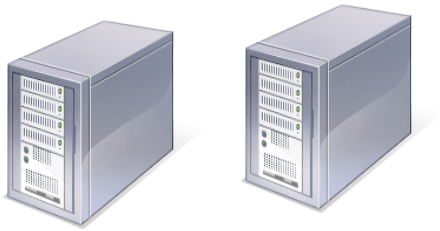


Resource identifier

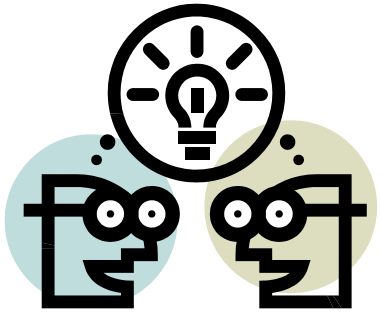
Resolver

?

Service endpoint identifier
Capabilities



Q&A



 12/05/23

Summary: Application protocols

- How do I send information?
  - Rely on presentation layer
- How do I interact with the other end?
  - You can roll your own protocol, but be cautious.
- How do I find the other end?
  - Three representative patterns

