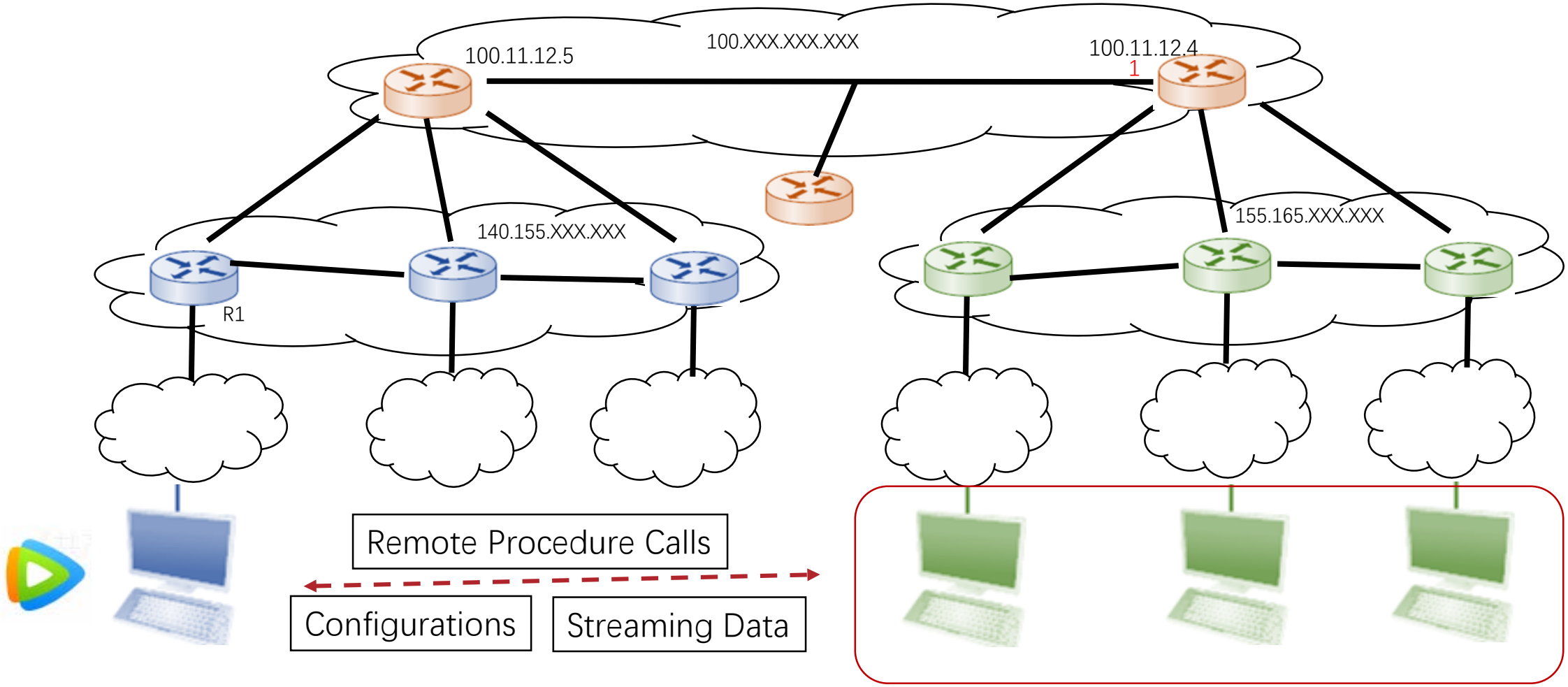
CS120: Computer Networks

Lecture 22. End-to-End Data 1

Zhice Yang

Data in End-to-End Connections

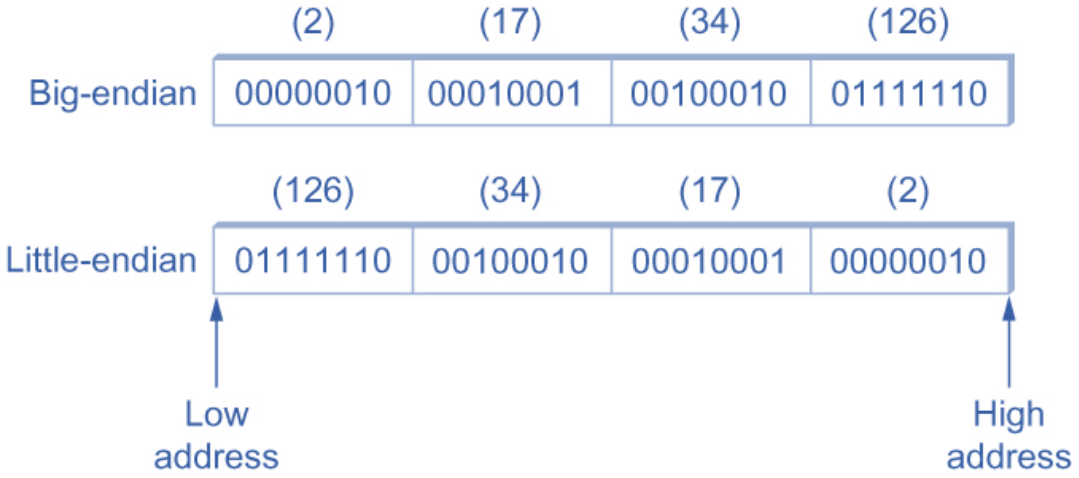


Data in End-to-End Connections

- Presentation Formatting
- Data Compression
  - Lossless Compression
  - Multimedia Compression

Presentation Formatting

- Challenges
  - Different Host Architecture: 16bit, 32bit, 64bit
    - eg. long
  - Different Compilers
    - Different layout/padding of structures
    - eg. struct BitField { unsigned char : 2; unsigned int : 2; }
  - Different base type representation
    - eg. 34,677,374



Presentation Formatting

- Solution
  - Marshalling (encoding) application data into messages
  - Unmarshalling (decoding) messages into application data

Presentation Formatting

- Solution
  - Conversion Strategy
    - Canonical intermediate form
    - Receiver-makes-right
  - Base types (e.g., ints, floats) => Convert
  - Flat types (e.g., structures, arrays) => Pack to base types
  - Complex types (e.g., pointers) => Serialization

Presentation Formatting: Examples

- eXternal Data Representation (XDR)
  - Used in SunRPC
  - Canonical intermediate form
  - Defined in RFC1014
    - C-type
    - big-endian
    - Step in 4-bytes
    - etc.

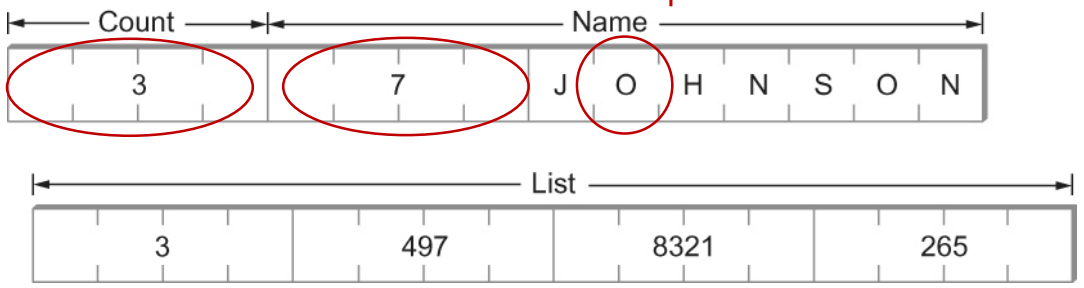
Presentation Formatting: Examples

- eXternal Data Representation (XDR) Steps:
  - Define bytes to be serialized in struct
  - Compile in client and server
  - Stub helps to encode and decode

```
#define MAXNAME 256;
#define MAXLIST 100;
struct item {
 int count;
 char name[MAXNAME];
 int list[MAXLIST];
};
bool_t
xdr_item(XDR *xdrs, struct item *ptr)
{
 return(xdr_int(xdrs, &ptr->count) &&
         xdr_string(xdrs, &ptr->name, MAXNAME) &&
         xdr_array(xdrs, &ptr->list, &ptr->count,
                   MAXLIST, sizeof(int), xdr_int));
}
```

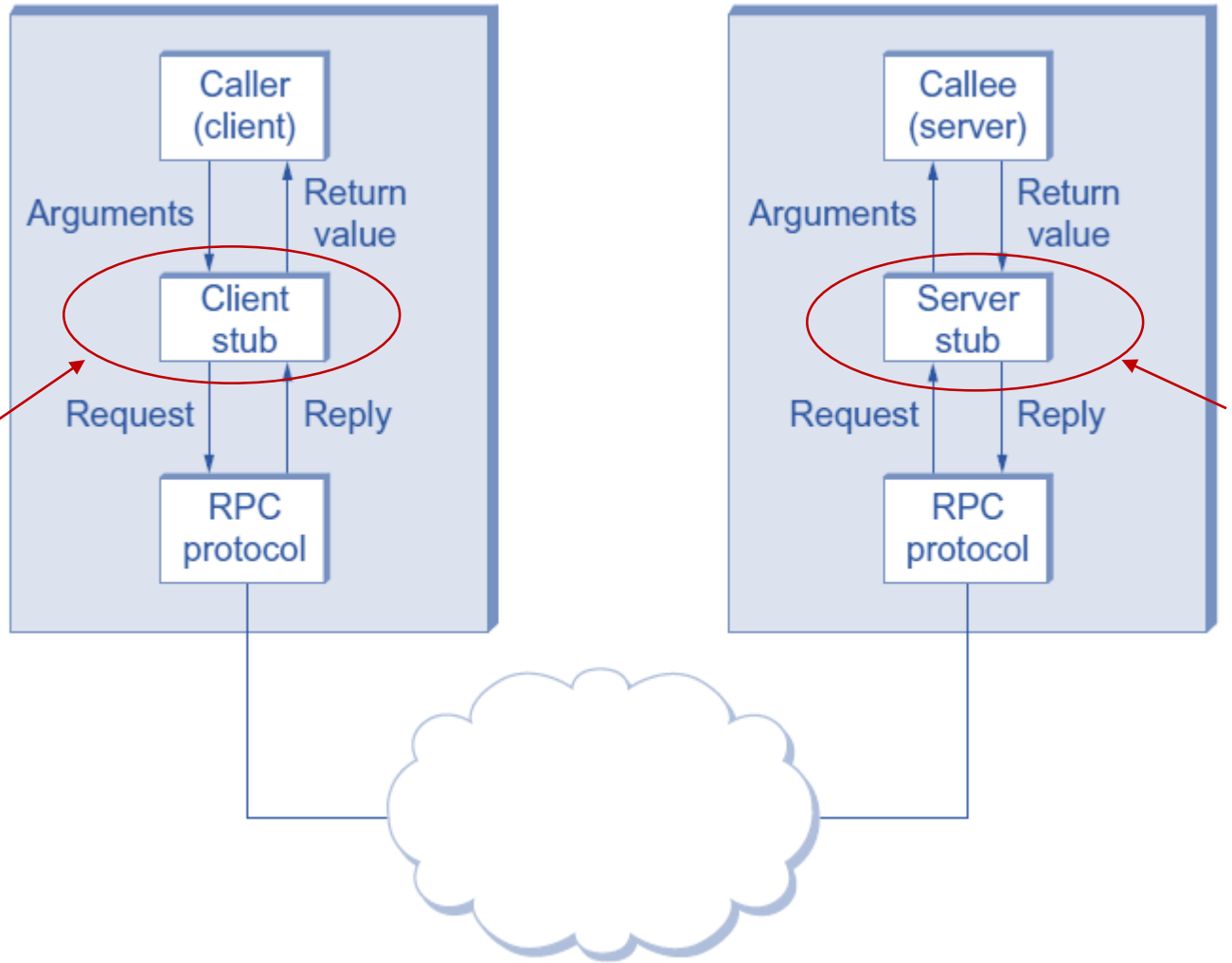
Size of each element is a multiple of 4 bytes

Except char



RPC Mechanism

Stub is like a proxy to translates procedure calls between network transmissions



Marshals parameters and calls the server

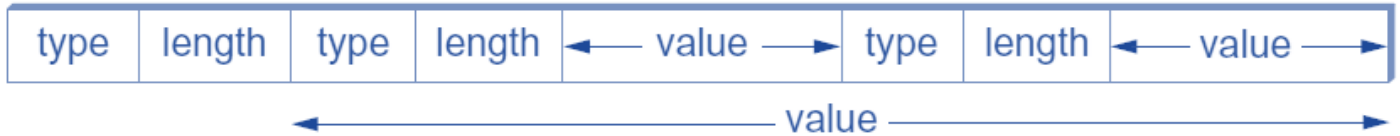
Unmarshals parameters and calls the local function

Presentation Formatting: Examples

- Abstract Syntax Notation One (ASN.1)
  - ISO Standard, used in SNMP
  - Canonical intermediate form
  - Based on tag: <tag, length, value>
  - Format can be interpreted, but of low efficiency

Integer:

Compound types:

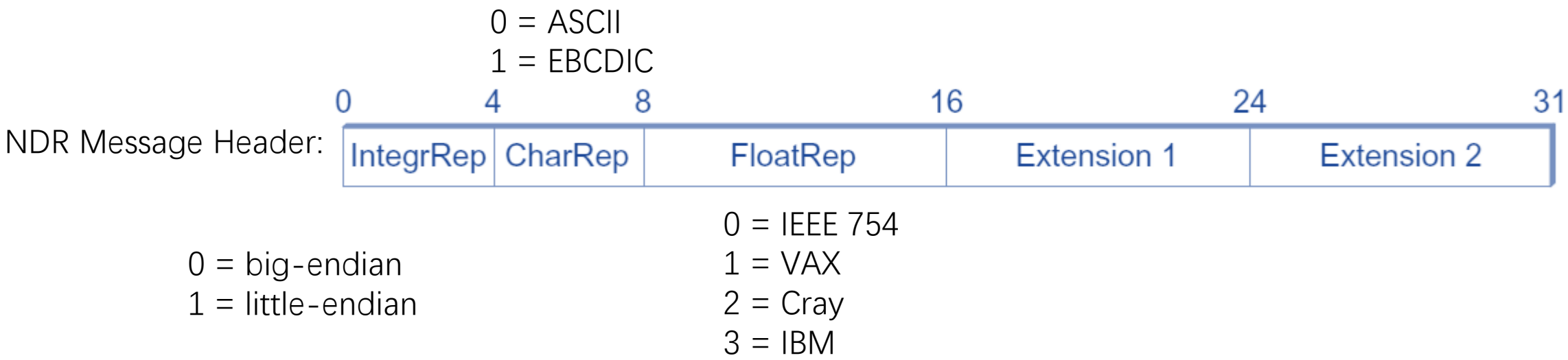


Length Field for Long Data:

Length Field for Shot Data:

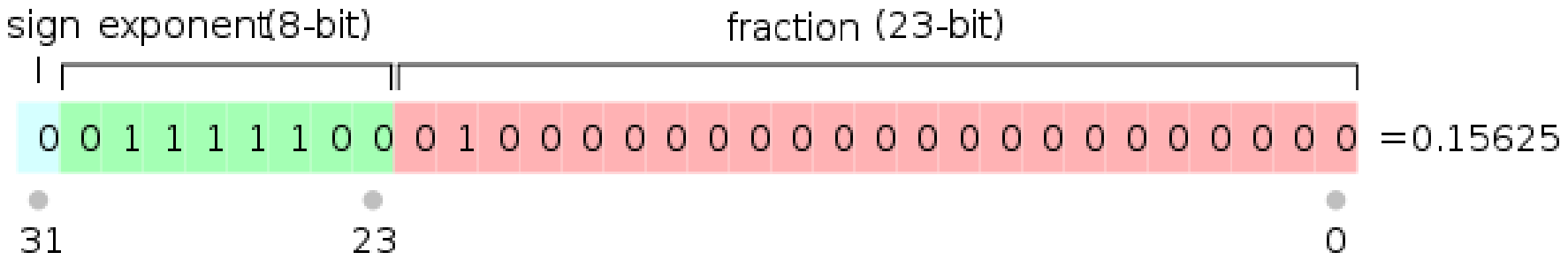
Presentation Formatting: Examples

- Network Data Representation (NDR)
  - Used in DCE
  - Receiver-makes-right
  - Architecture tag at the front of each message

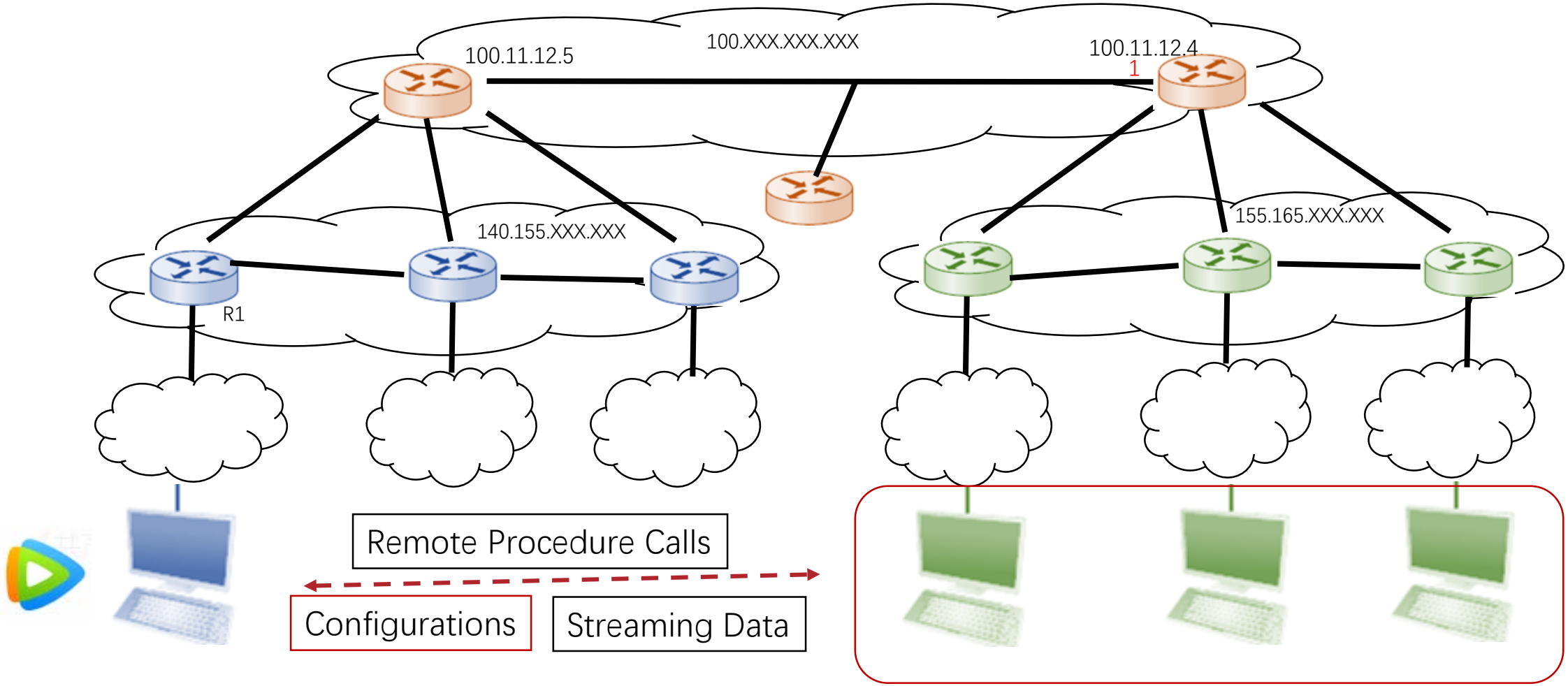


IEEE 754

- https://www.h-schmidt.net/FloatConverter/IEEE754.html



Data in End-to-End Connections



Markup Languages

- Examples: XML and HTML
- Approach
  - Data is represented as text
    - Readable for human
    - Can reuse parsers
  - Text tags (markup) are used to express information about the data.

```
<?xml version="1.0"?>
<employee>
        <name>John Doe</name>
        <title>Head Bottle Washer</title>
        <id>123456789</id>
        <hiredate>
                <day>5</day>
                <month>June</month>
                <year>1986</year>
        </hiredate>
</employee>
```

Markup

Nested structure

Extensible Markup Language (XML)

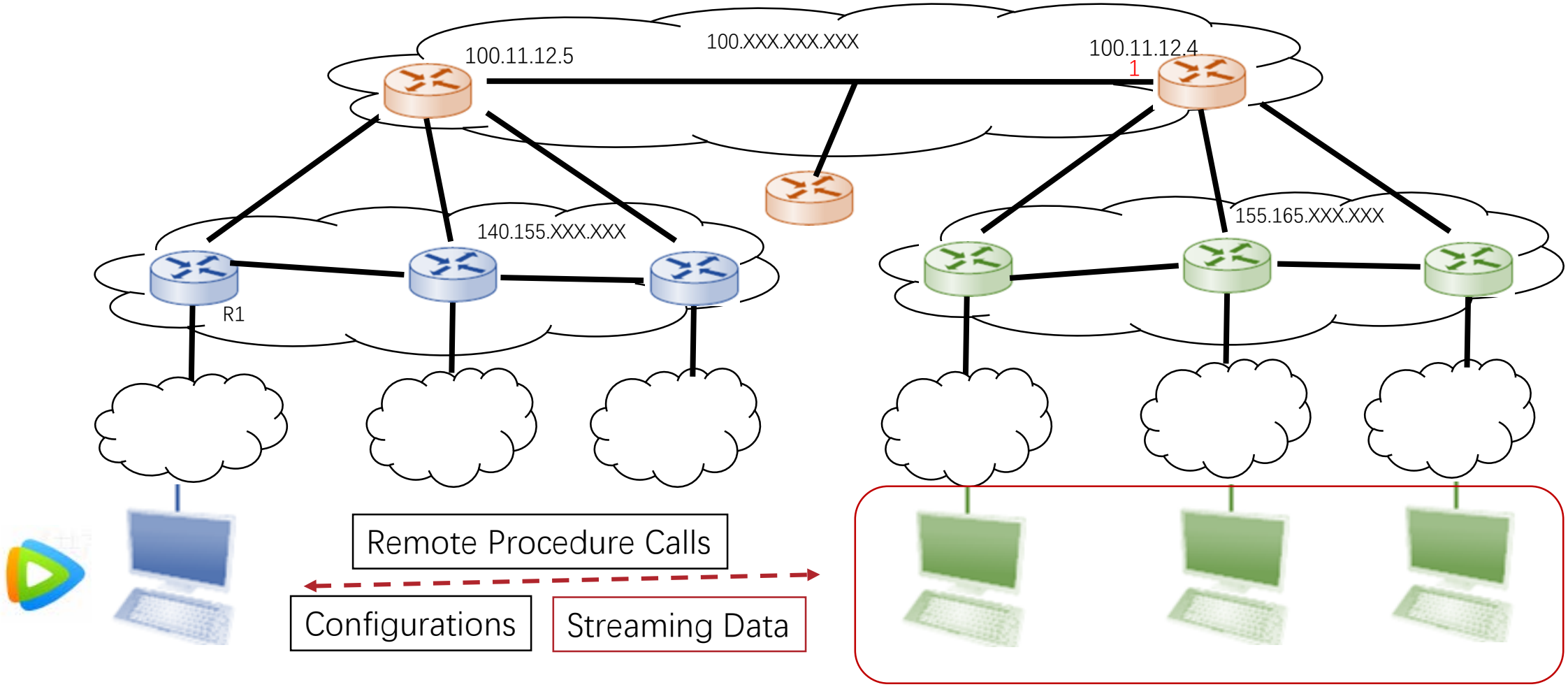
- XML Schema
  - Define XML

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
   <element name="employee">
       <complexType>
           <sequence>
               <element name="name" type="string"/>
               <element name="title" type="string"/>
               <element name="id" type="string"/>
                  <element name="hiredate">
                     <complexType>
                         <sequence>
                            <element name="day" type="integer"/>
                            <element name="month" type="string"/>
                            <element name="year" type="integer"/>
                         </sequence>
…
```

Extensible Markup Language (XML)

- XML Namespace
  - Use Uniform Resource Identifier (URL) to identify a unique namespace
  - Define an XML namespace
    - `xmlns:emp="http://www.example.com/employee">`
  - Identifier with namespace
    - `<emp:title>Head Bottle Washer</temp:title>`

Data in End-to-End Connections



Traffic of a Full Size Video Stream

- Resolution: 1920*1080
- Framerate: 30fps
- Color per pixel: 3
- Color depth: 8 bits
- Required Throughput: 1920*1080*3*8*30 bps = 1.5Gbps

Data in End-to-End Connections

- Data Presentation
- ➢Data Compression
  - Lossless Compression
  - Multimedia Compression

gzip

- GNU zip
- A widely-used lossless compression method
- Main Algorithms
  - LZ Algorithm
  - Huffman Coding

LZ Algorithm

- Dictionary-Based Compression
- Method
  - Construct dictionary: find repeated strings
  - Repeated strings are represented by its index in dictionary
    - eg. Repeated strings are simplified to <distance, length> pair
      - blah blah b! => blah [D=5, L=6]!

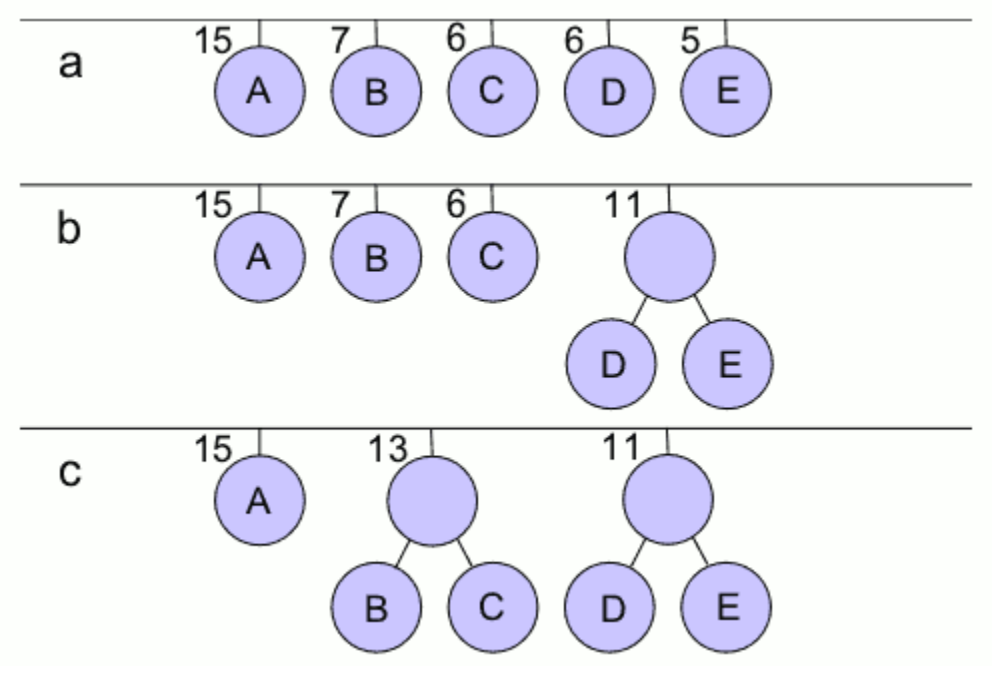
Huffman Coding

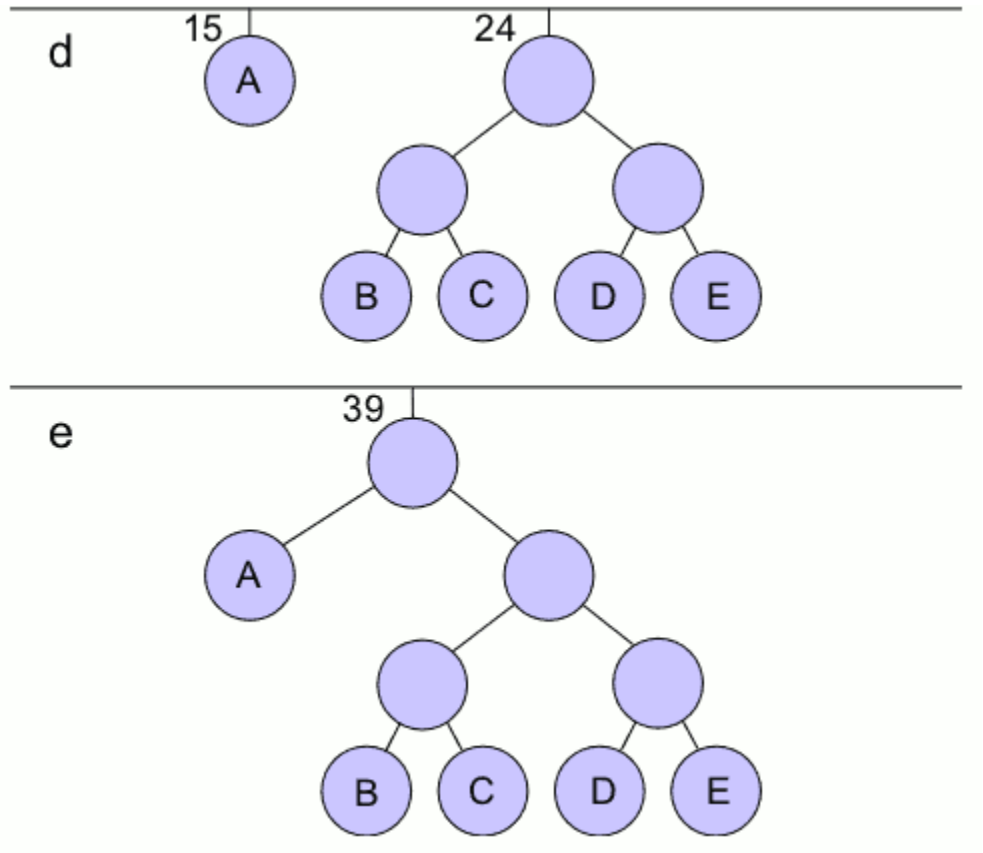
- Intuition: Higher frequency characters => less bit to representation
- A:90%, B:5%, C:5%
  - A: 1
  - B: 01
  - C: 00

Huffman Coding

- Create a leaf node for each symbol and add it to the priority queue.
- While there is more than one node in the queue:
  - Remove the two nodes of highest priority (lowest probability) from the queue
  - Create a new internal node with these two nodes as children and with probability equal to the sum of the two nodes' probabilities.
  - Add the new node to the queue.
- The remaining node is the root node and the tree is complete.

Huffman Coding





ref: https://en.wikipedia.org/wiki/Huffman_coding

bzip2

- Open-source Compression Method
- Better Compression ratio than gzip
- Main Components
  - Run Length Encoding (RLE)
  - Burrows–Wheeler Transform (BTW)
  - Move to Front (MTF)
  - Huffman Coding

Run Length Encoding

- Basic idea:
  - Replace consecutive occurrences of a given symbol with only one copy of the symbol
  - Plus a count of how many times that symbol occurs
  - eg.: AAABBCDDDD => 3A2B1C4D

Burrows–Wheeler Transform

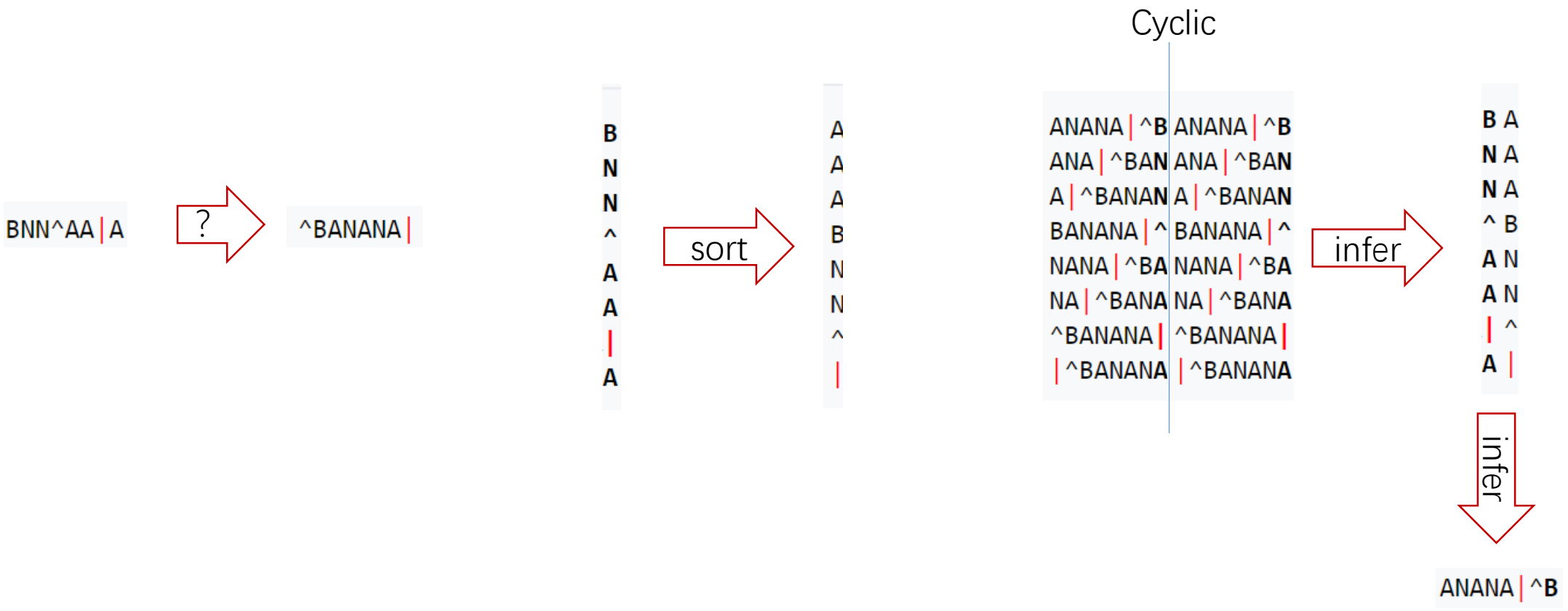
- Basic idea:
  - Move same characters together

Transformation				
Input	All rotations	Sorted into lexical order	Taking last column	Output last column
^BANANA|	^BANANA| |^BANANA A|^BANAN NA|^BANA ANA|^BAN NANA|^BA ANANA|^B BANANA|^	ANANA|^B ANA|^BAN A|^BANAN BANANA|^ NANA|^BA NA|^BANA ^BANANA| |^BANANA	ANANA|^B ANA|^BAN A|^BANAN BANANA|^ NANA|^BA NA|^BANA ^BANANA| |^BANANA	BNN^AA|A

ref: https://en.wikipedia.org/wiki/Burrows%E2%80%93Wheeler_transform

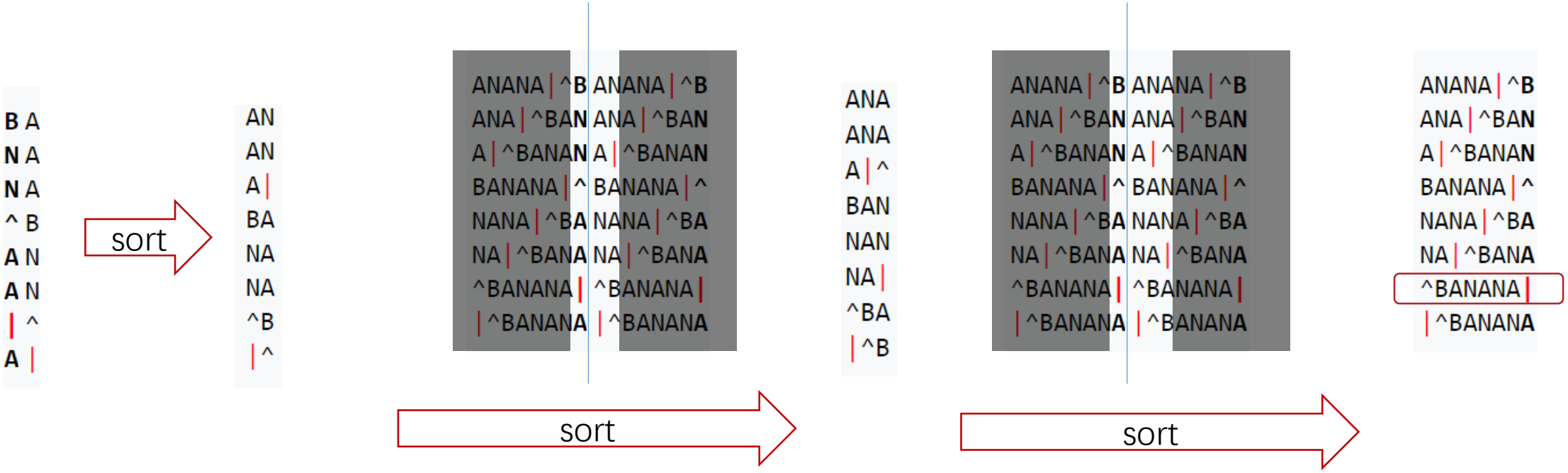
Burrows–Wheeler Transform

- Recovery



Burrows–Wheeler Transform

- Recovery Algorithm



Move to Front

- Transform successive same characters to 0 from BWT
- Improve the effectiveness of Huffman coding

ccbbbaaaaa	…abc… (ASCII)	[**99**]
c**c**bbbaaaaa	c…ab…	[99,**0**]
cc**b**bbaaaaa	c…ab…	[99,0,**99**]
ccb**b**baaaaa	bc…a…	[99,0,99,**0**]
ccbb**b**aaaaa	bc…a…	[99,0,99,0,**0**]
ccbbb**a**aaaa	bc…a…	[99,0,99,0,0,**99**]
ccbbba**a**aaa	abc…	[99,0,99,0,0,99,**0**]
ccbbbaa**a**aa	abc…	[99,0,99,0,0,99,0,**0**]
ccbbbaaa**a**a	abc…	[99,0,99,0,0,99,0,0,**0**]
ccbbbaaaa**a**	abc…	**[99,0,99,0,0,99,0,0,0,0]**

Run Length Encoding

- Compress 0s from MTF
- Use special binary symbol: RUNA and RUNB

Reference

- Textbook 7.1 7.2