Application Layer

Networking
Marquette University 2017

DNS

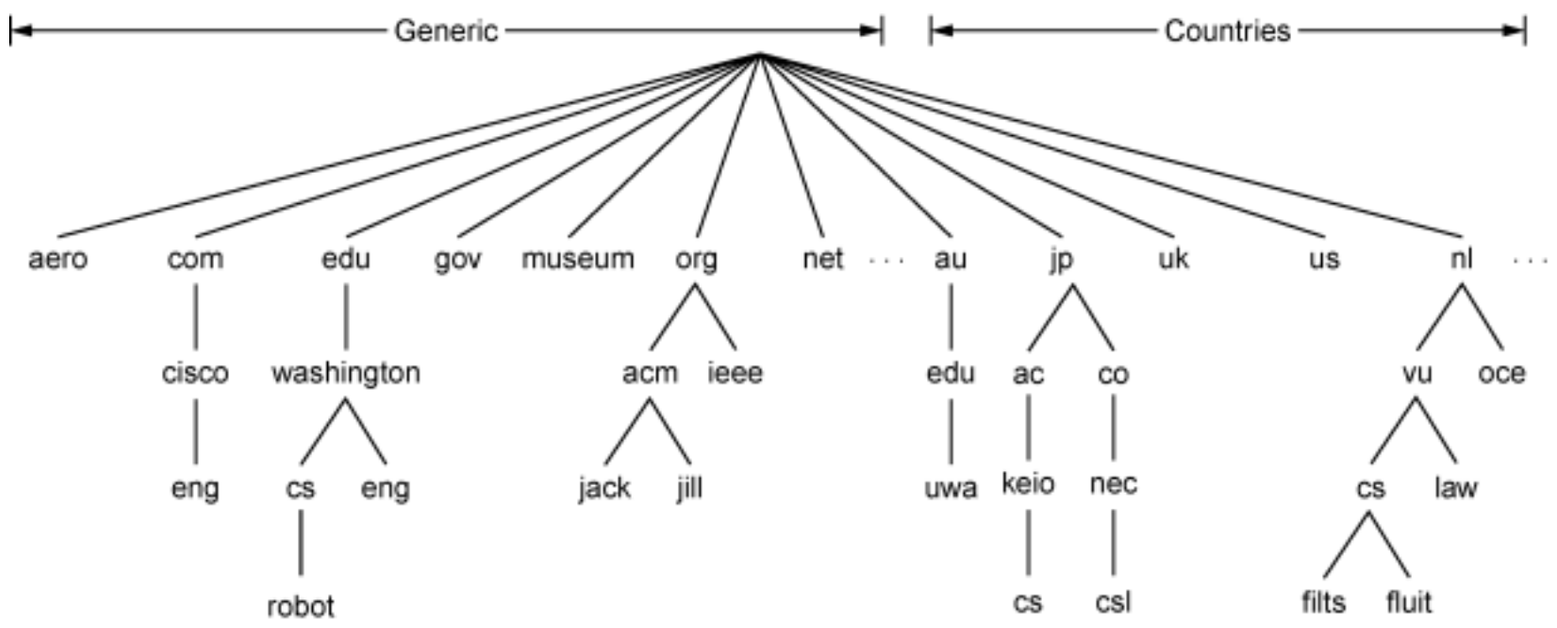
- Domain Name Service
  - Binds human readable names to internet addresses
    - Marquette University <—> 134.48.29.23
  - History:
    - Before DNS:
      - /etc/hosts on Linux
      - C:\Windows\System32\drivers\etc\hosts
    - Changes were submitted to Network Information Center (SRI) via email

DNS

- /hosts fell apart:
  - Single entity could not handle the load
  - Names were difficult:
    - USF stands for: University of Southern Florida, University of San Francisco
- 1983: DNS was born

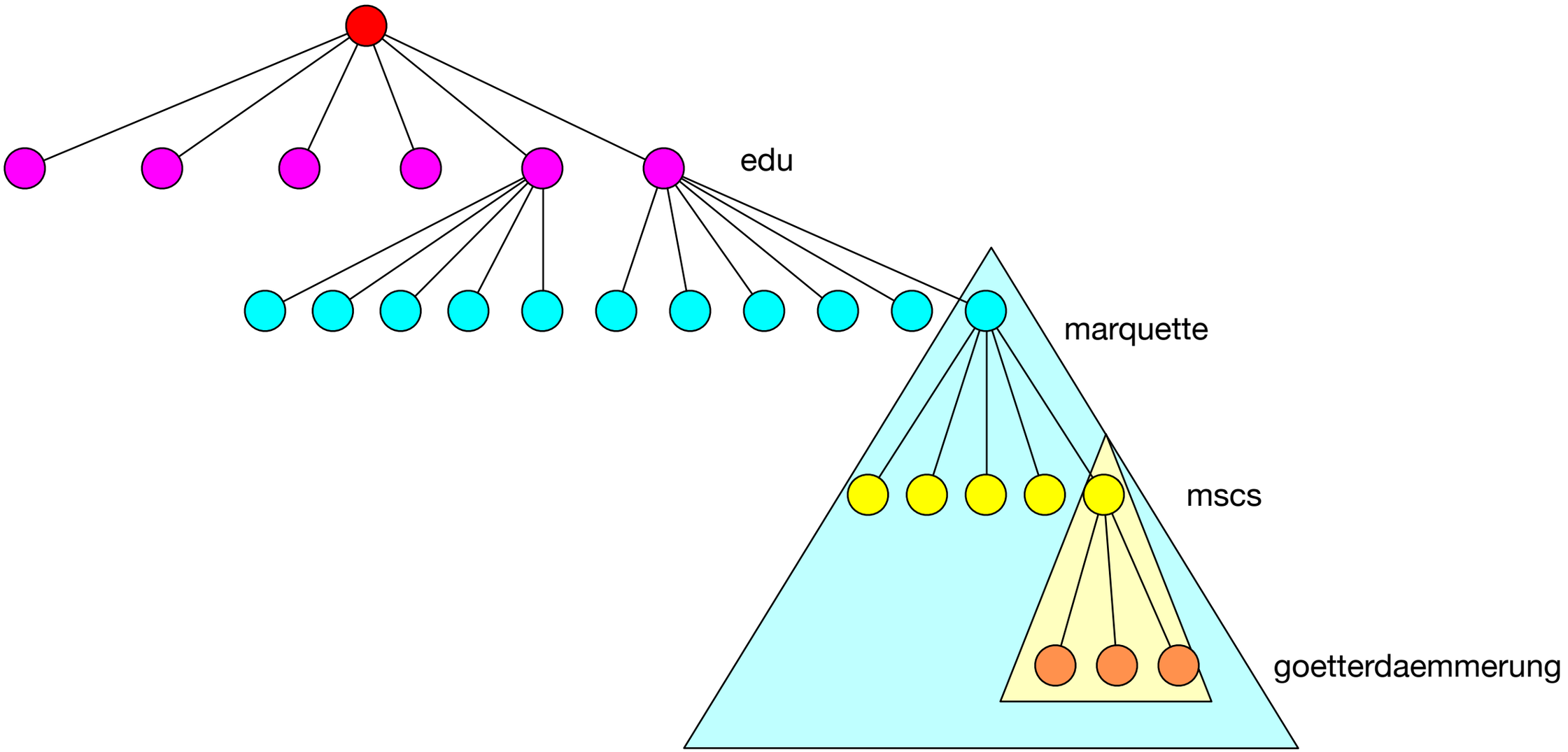
DNS

- Domain Name System
  - Hierarchical, domain-based naming scheme
  - Distributed database system
  - Maps host names to IP numbers



Domain	Intended use	Start date	Restricted?
com	Commercial	1985	No
edu	Educational institutions	1985	Yes
gov	Government	1985	Yes
int	International orgs	1988	Yes
mil	Military	1985	Yes
net	Network providers	1985	No
org	Non-profit organizations	1985	No
aero	Air transport	2001	Yes
biz	Businesses	2001	No
coop	Cooperatives	2001	Yes
info	Informational	2002	No
museum	Museums	2002	Yes
name	People	2002	No
pro	Professionals	2002	Yes
cat	Catalan	2005	Yes
jobs	Employment	2005	Yes
mobi	Mobile devices	2005	Yes
tel	Contact details	2005	Yes
travel	Travel industry	2005	Yes
xxx	Sex industry	2010	No

DNS



goetterdaemmerung.mscs.marquette.edu

DNS

- Tree is divided into zones
  - Each zone has an administrator
  - Responsible for that domain lies with the administrator
- E.g.: .com
  - Administered by the US Department of Defense
  - Today operated by Verisign
    - Verisign registrations in .com are processed via registrars accredited by ICANN
- E.g.: .edu
  - Administered by the US Department, but now sub-contracted to educause

Fun Fact

- The 92nd oldest entry in .edu is mu.edu for Marquette University
- But beaten by several month by Georgetown University

DNS

- Functions of each DNS server:
  - Authority over a portion of the hierarchy
    - No need to store all DNS names
    - Store all the records for hosts/domains in its zone
    - May be replicated for robustness
    - Know the addresses of the root servers
      - Resolve queries for unknown names
        - Root servers know about all TLDs
  - The buck stops at the root servers

Root Name Servers

- Responsible for “root” zone
  - Approx. 13 root name servers worldwide
  - Same name and IP address, but all distributed
  - Currently {a-m}.root-servers.net
  - Local name servers contact root servers when they cannot resolve a name

Root Name Servers

a.root-servers.net	198.41.0.4, 2001:503:ba3e::2:30	Verisign, Inc.
b.root-servers.net	199.9.14.201, 2001:500:200::b	University of Southern California, Information Sciences Institute
c.root-servers.net	192.33.4.12, 2001:500:2::c	Cogent Communications
d.root-servers.net	199.7.91.13, 2001:500:2d::d	University of Maryland
e.root-servers.net	192.203.230.10, 2001:500:a8::e	NASA (Ames Research Center)
f.root-servers.net	192.5.5.241, 2001:500:2f::f	Internet Systems Consortium, Inc.
g.root-servers.net	192.112.36.4, 2001:500:12::d0d	US Department of Defense (NIC)
h.root-servers.net	198.97.190.53, 2001:500:1::53	US Army (Research Lab)
i.root-servers.net	192.36.148.17, 2001:7fe::53	Netnod
j.root-servers.net	192.58.128.30, 2001:503:c27::2:30	Verisign, Inc.
k.root-servers.net	193.0.14.129, 2001:7fd::1	RIPE NCC
l.root-servers.net	199.7.83.42, 2001:500:9f::42	ICANN
m.root-servers.net	202.12.27.33, 2001:dc3::35	WIDE Project

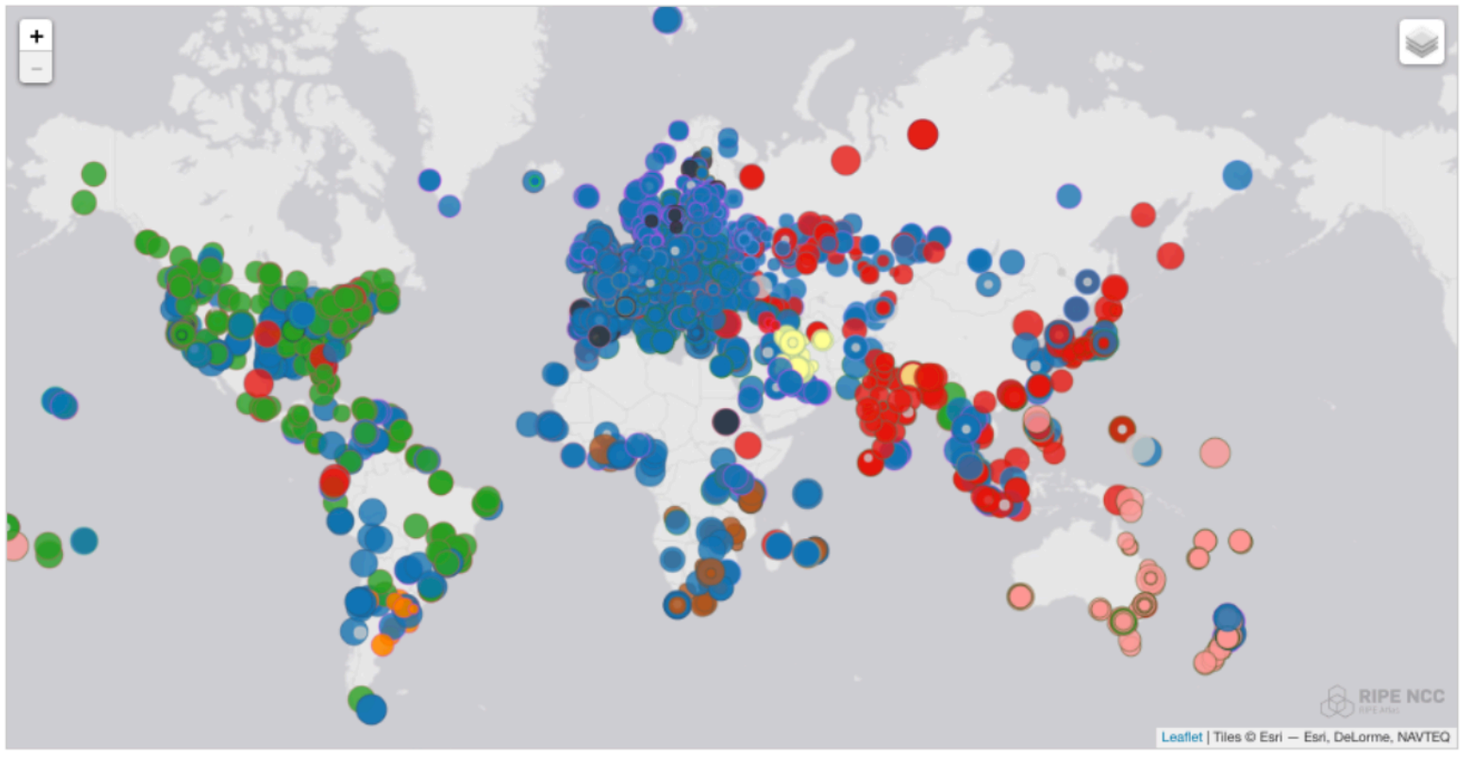
- Information on each root name server is available at x.root-servers.org

Root Name Servers

- A query to a root-server is an **any-cast**
  - **Any-cast** is implemented by the BGP
  - RFC 4892 (June 2007) allows identifying the location of the root server

Root Name Servers

- Example: k-root server run by RIPE NCC visible from around the world



k-root server seen from different places:
Blue: European k-roots (Amsterdam, London, Stockholm)
Green: North America, Orange: South America,
Brown: Africa, Red: South Asia, Pink: Australia and Oceania

Root Name Servers

- Each k-node announces prefixes from 193.0.14.0/23 in AS25152
  - Authoritative DNS announce 193.0.9.0/24 and 2001:67c:e0::/48 from AS 197000
- Every-one can apply to run an instance

Top Level Domain Servers

- Top-level domain servers (TLD)
  - Responsible for maintaining records mapping IP addresses
    - Limited to top-level domains
      - Country domains: uk, fr, us, ca, …
      - .com, .org, .net, .edu, …
        - As seen: educause for .edu, Verisign for .com,

Authoritative DNS Servers

- Authoritative DNS Servers
  - Organizations own DNS server
    - Provides authoritative hostname to IP mapping
      - for organization’s named host
  - Maintained by organization or Internet service provider

Local DNS Server

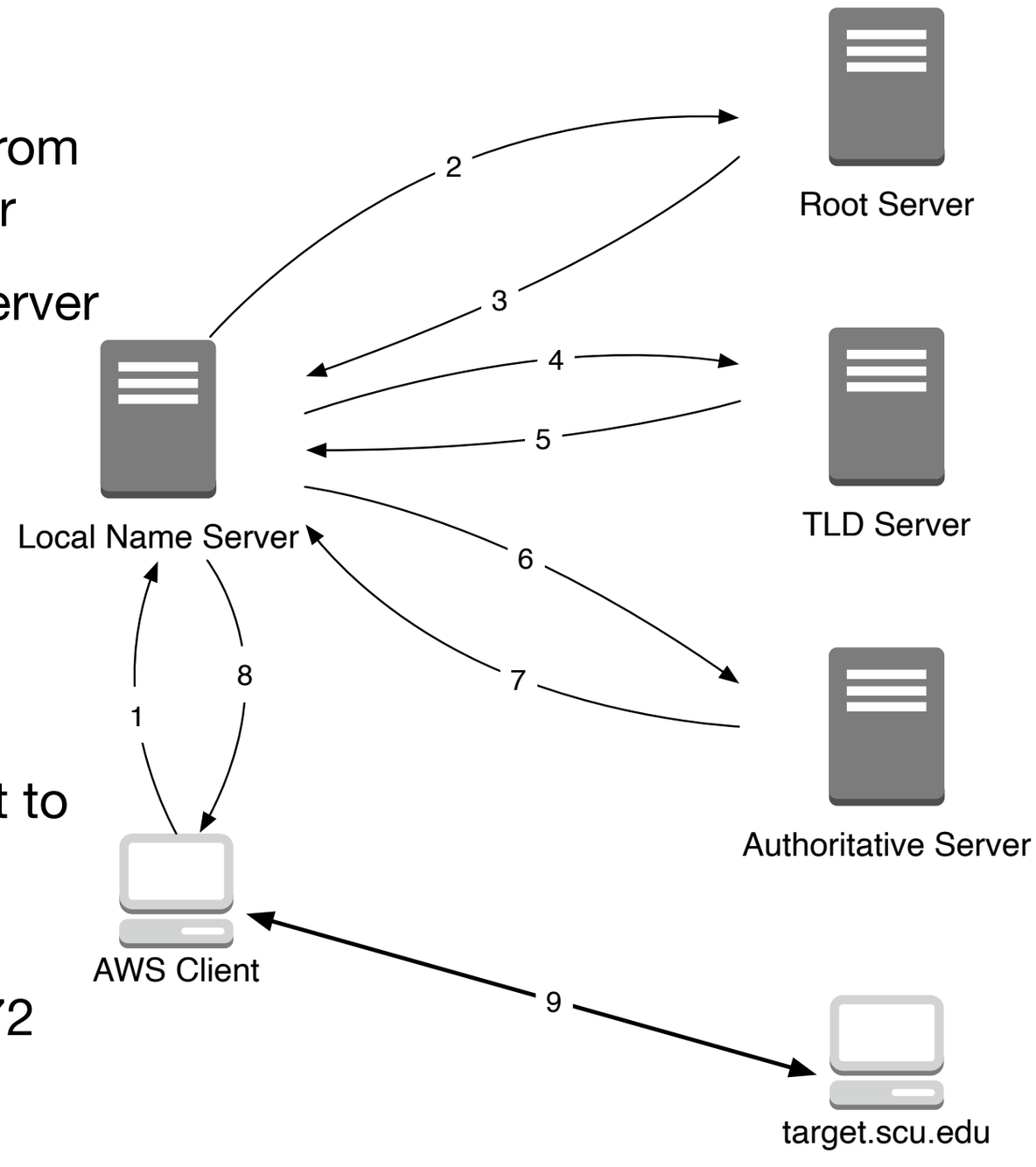
- When a host makes a DNS query
  - Query is sent to its local DNS server
  - Set-up e.g. with DHCP
- Local DNS server has:
  - Local cache of recent translation pairs
    - Might be out of date
  - Acts as proxy: forwards into the hierarchy

DNS Name Resolution

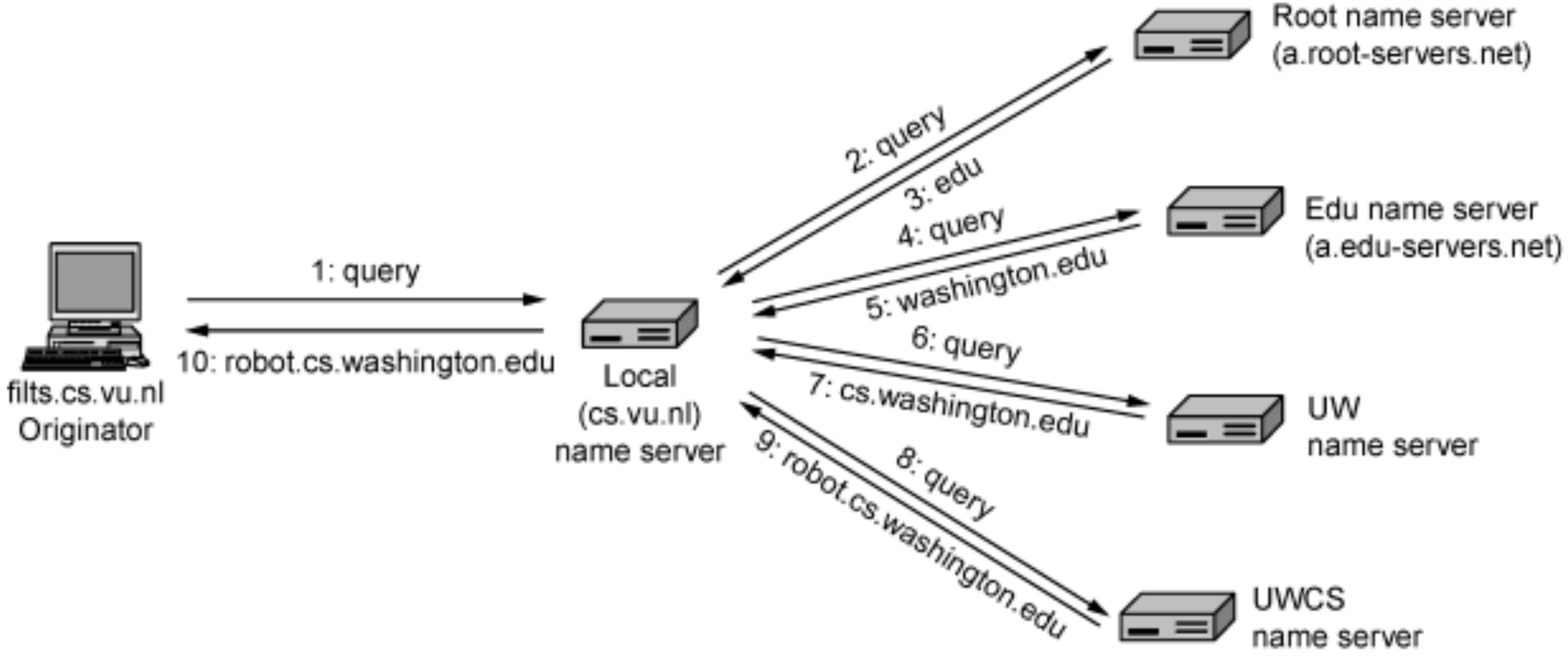
- DNS lookups
  - Single name server is not possible
  - Use local information
  - Requests go always to the local name server
  - Local name server then resolves using root-server and subordinate servers

DNS Name Resolution

1. Client contacts local DNS server
2. Local DNS server serves IP address from cache, otherwise: contacts root server
3. Root server returns address of TLD server
4. Local DNS server asks TLD server
5. TLD server returns address of Authoritative Server
6. Local DNS server asks Authoritative Server
7. Authoritative Server sends IP of target to Local DNS server
8. Local DNS server informs client of IP address (and caches result for up to 72 hours)
9. Client now connects to target



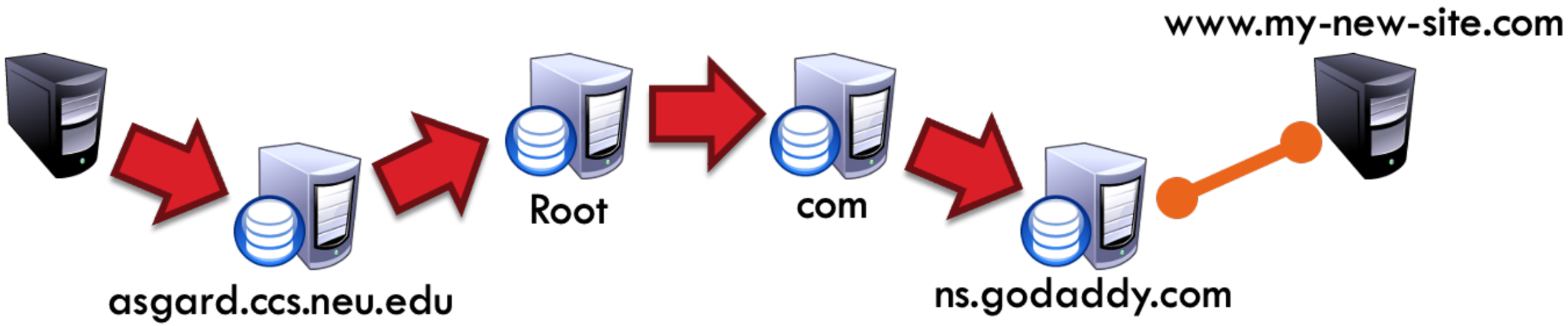
DNS: Record Lookup



Example of a resolver looking up a remote name in 10 steps.

DNS

- New Domain Name:
  - Needs to propagate through the various caches



DNS: Domain Resource Records

- Every domain has *resource records* associated with it
  - Domain_name, Time_to_live, Class, Type, Value

Type	Meaning	Value
SOA	Start of authority	Parameters for this zone
A	IPv4 address of a host	32-Bit integer
AAAA	IPv6 address of a host	128-Bit integer
MX	Mail exchange	Priority, domain willing to accept email
NS	Name server	Name of a server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
SPF	Sender policy framework	Text encoding of mail sending policy
SRV	Service	Host that provides it
TXT	Text	Descriptive ASCII text

DNS: Dig Example

- Can use dig to obtain DNS information
  - Example: `dig marquette.edu MX`
    - DNS query for marquette email server
    - Answer:

```
;; ANSWER SECTION:
marquette.edu.      600  IN MX 0 marquette-edu.mail.protection.outlook.com.
```

DNS

```
; Authoritative data for cs.vu.nl
cs.vu.nl.       86400   IN      SOA     star boss (9527,7200,7200,241920,86400)
cs.vu.nl.       86400   IN      MX      1 zephyr
cs.vu.nl.       86400   IN      MX      2 top
cs.vu.nl.       86400   IN      NS      star

star            86400   IN      A       130.37.56.205
zephyr          86400   IN      A       130.37.20.10
top             86400   IN      A       130.37.20.11
www             86400   IN      CNAME   star.cs.vu.nl
ftp             86400   IN      CNAME   zephyr.cs.vu.nl

flits           86400   IN      A       130.37.16.112
flits           86400   IN      A       192.31.231.165
flits           86400   IN      MX      1 flits
flits           86400   IN      MX      2 zephyr
flits           86400   IN      MX      3 top

rowboat                 IN      A       130.37.56.201
                        IN      MX      1 rowboat
                        IN      MX      2 zephyr

little-sister           IN      A       130.37.62.23

laserjet                IN      A       192.31.231.216
```

DNS Attacks

- Distributed Denial of Service Attack on DNS servers
  - Attacker sends spoofed queries to third-party open resolvers
    - Queries result in very large results
    - Sent to the victim

DNS Attacks

- Cache Poisoining
  - DNS grabs all (almost all, some) responses and puts into cache
  - Just send false DNS responses to where a DNS server can pick them up
- This one is pretty much shut down, BUT

DNS Attacks

- Cache Poisoning
  - Attacker queries a recursive name server for IP address of a malicious site
  - Recursive name server queries **malicious** DNS resolver
  - Malicious resolver provides requested rogue IP address
    - **and also maps the rogue IP address to additional legitimate sites (us.bank.com)**
  - Any user querying for the IP address of the legitimate site gets the rogue IP address

DNS Attacks

- Use open port 53 (DNS queries)
  - for TCP SYN Floods
  - for DNS tunneling (covert communication channel)

DNS Attacks

- DNS hijacking
  - Modifies DNS record setting (e.g. at the domain registrar) to point to a rogue DNS server or domain

DNS Attack

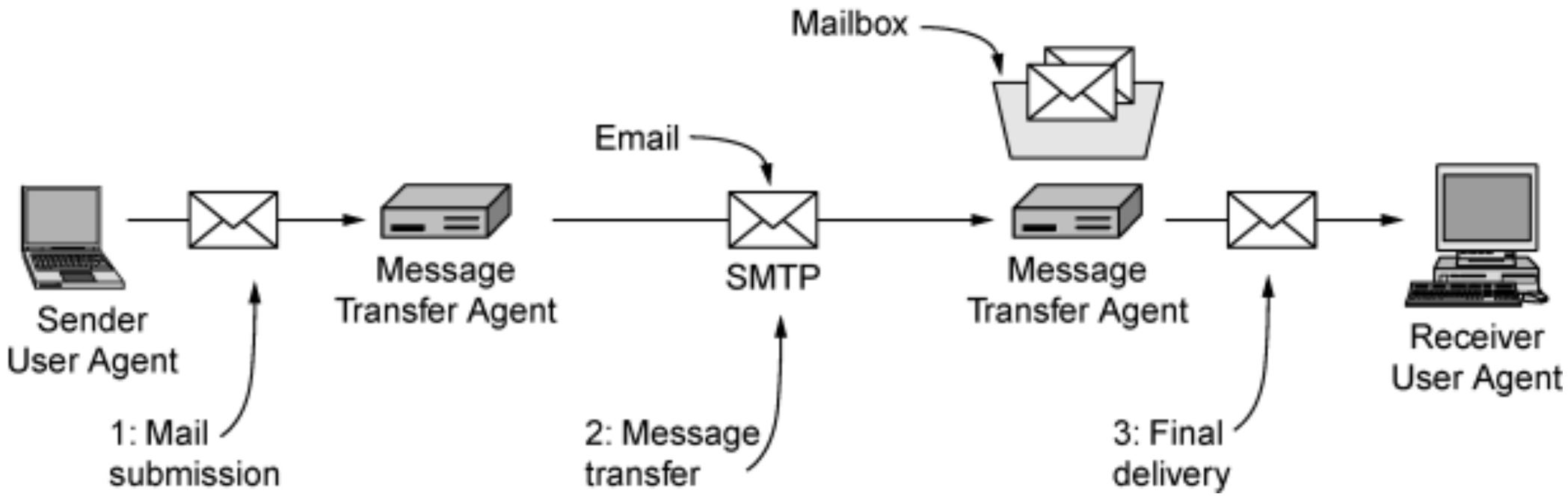
- NXDOMAIN attack
  - Attacker floods DNS server with queries to a non-existent domain / domain name
  - Fills up cache with NXDOMAIN results

DNSSEC

- DNSSEC provides
  - message origin authentication
  - message integrity
- DNSSEC **does not** provide confidentiality
  - DNS traffic can be intercepted to find out **where** you are going.

E-Mail

- Email uses User and Message Transfer Agents



Architecture of the email system.

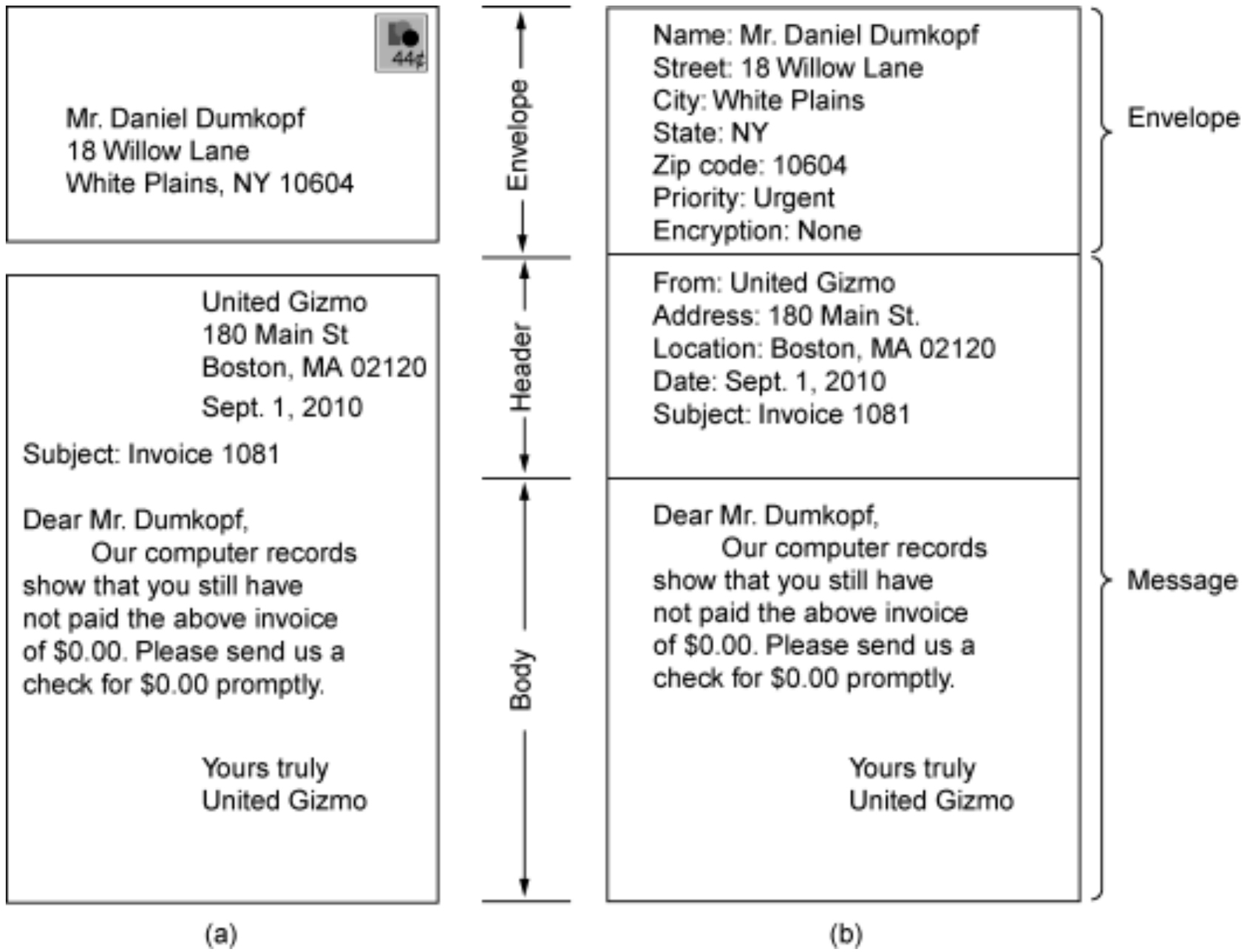
E-Mail

- Message transfer agents
  - Usually use Simple Mail Transfer Protocol
    - RFC 821, 5321

E-Mail

- Envelope
  - Information necessary to transport the message
  - Message, consisting of
    - Header
    - Body

E-Mail


44¢
Mr. Daniel Dumkopf
18 Willow Lane
White Plains, NY 10604
United Gizmo
180 Main St
Boston, MA 02120
Sept. 1, 2010
Subject: Invoice 1081
Dear Mr. Dumkopf,
Our computer records show that you still have not paid the above invoice of $0.00. Please send us a check for $0.00 promptly.
Yours truly
United Gizmo
(a)
Envelope
Header
Body
Name: Mr. Daniel Dumkopf
Street: 18 Willow Lane
City: White Plains
State: NY
Zip code: 10604
Priority: Urgent
Encryption: None
From: United Gizmo
Address: 180 Main St.
Location: Boston, MA 02120
Date: Sept. 1, 2010
Subject: Invoice 1081
Dear Mr. Dumkopf,
Our computer records show that you still have not paid the above invoice of $0.00. Please send us a check for $0.00 promptly.
Yours truly
United Gizmo
Envelope
Message
(b)

E-Mail

Header	Meaning
To:	Email address(es) of primary recipient(s)
Cc:	Email address(es) of secondary recipient(s)
Bcc:	Email address(es) for blind carbon copies
From:	Person or people who created the message
Sender:	Email address of the actual sender
Received:	Line added by each transfer agent along the route
Return-Path:	Can be used to identify a path back to the sender

RFC 5322 header fields related to message transport.

E-Mail

Header	Meaning
Date:	The date and time the message was sent
Reply-To:	Email address to which replies should be sent
Message-Id:	Unique number for referencing this message later
In-Reply-To:	Message-Id of the message to which this is a reply
References:	Other relevant Message-Ids
Keywords:	User-chosen keywords
Subject:	Short summary of the message for the one-line display

Some fields used in the RFC 5322 message header.

E-Mail

- Multipurpose Internet Mail Extensions
  - Add additional structure and encoding rules for non-ASCII messages

Header	**Meaning**
MIME-Version:	Identifies the MIME version
Content-Description:	Human-readable string telling what is in the
Content-Id:	Unique identifier
Content-Transfer-Encoding:	How the body is wrapped for transmission
Content-Type:	Type and format of the content

E-Mail

Type	Example subtypes	Description
text	plain, html, xml, css	Text in various formats
image	gif, jpeg, tiff	Pictures
audio	basic, mpeg, mp4	Sounds
video	mpeg, mp4, quicktime	Movies
model	vrml	3D model
application	octet-stream, pdf, javascript, zip	Data produced by applications
message	http, rfc822	Encapsulated message
multipart	mixed, alternative, parallel, digest	Combination of multiple types

MIME content types and example subtypes.

E-Mail: SMTP

- How to submit email
  - Insecure SMTP server

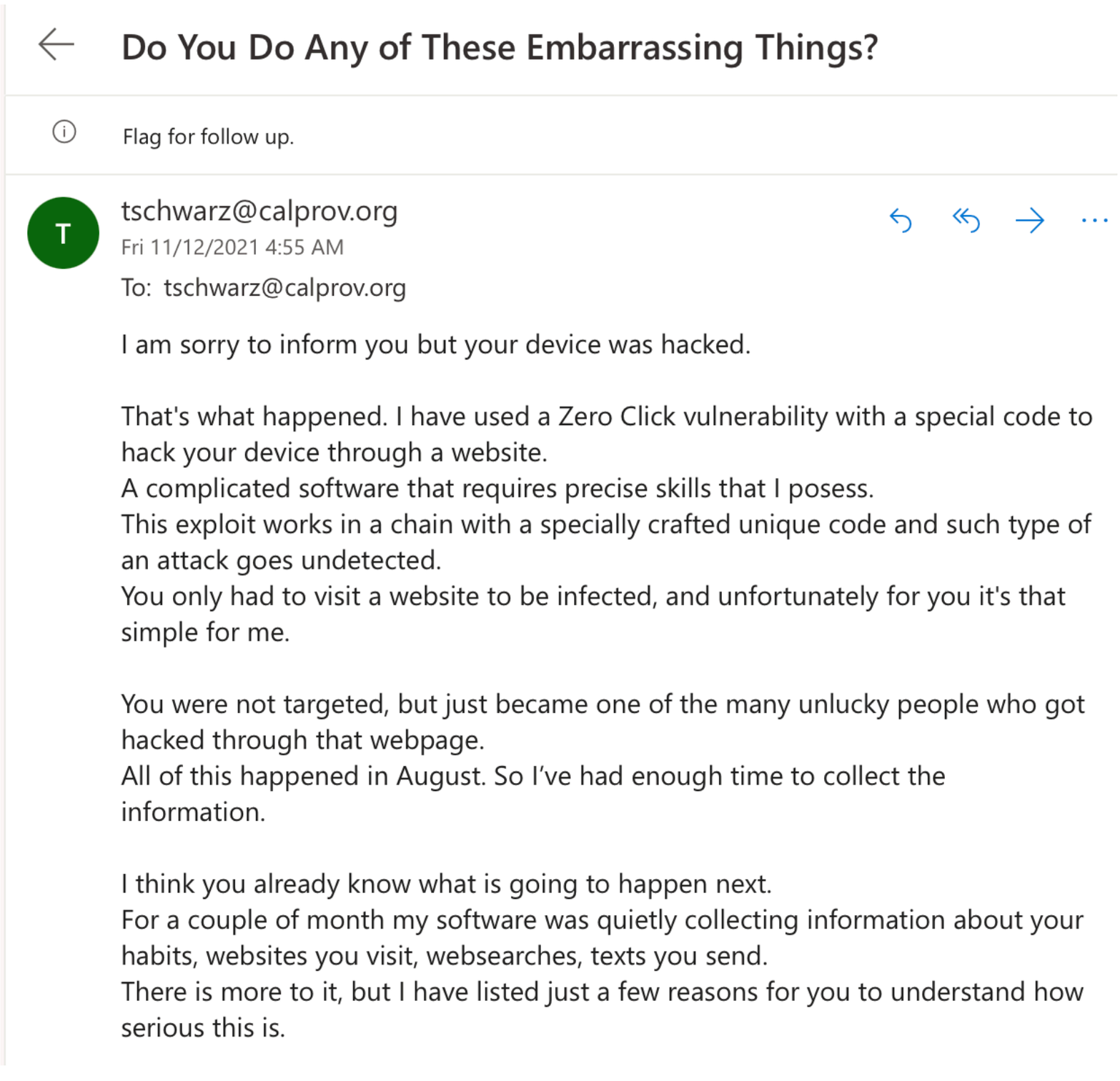
```
telnet server8.engr.scu.edu 25
220 server8.engr.scu.edu ESMTP Sendmail 8.12.10/8.12.10; Tue, 23 Dec 2003 16:32:
07 -0800 (PST)
helo 129.210.16.8
250 server8.engr.scu.edu Hello dhcp-19-198.engr.scu.edu [129.210.19.198], pleased to meet you
mail from: jholliday@engr.scu.edu
250 2.1.0 jholliday@engr.scu.edu... Sender ok
rcpt to: tschwarz
250 2.1.5 tschwarz... Recipient ok
data
354 Enter mail, end with "." on a line by itself
This is a spoofed message.
.
250 2.0.0 hBO0W76P002752 Message accepted for delivery
quit
221 2.0.0 server8.engr.scu.edu closing connection
```

E-Mail: SMTP

From jholliday@engr.scu.edu Tue Dec 23 16:44:55 2003
Return-Path: <jholliday@engr.scu.edu>
Received: from server8.engr.scu.edu (root@server8.engr.scu.edu [129.210.16.8])
by server4.engr.scu.edu (8.12.10/8.12.10) with ESMTP id hBO0itpv008140
for <tschwarz@engr.scu.edu>; Tue, 23 Dec 2003 16:44:55 -0800
From: JoAnne Holliday <jholliday@engr.scu.edu>
Received: from 129.210.16.8 (dhcp-19-198.engr.scu.edu [129.210.19.198])
by server8.engr.scu.edu (8.12.10/8.12.10) with SMTP id hBO0W76P002752
for tschwarz; Tue, 23 Dec 2003 16:41:55 -0800 (PST)
Date: Tue, 23 Dec 2003 16:32:07 -0800 (PST)
Message-Id: <200312240041.hBO0W76P002752@server8.engr.scu.edu>
X-Spam-Checker-Version: SpamAssassin 2.60-rc3 (1.202-2003-08-29-exp) on server4.engr.scu.edu
X-Spam-Level:
X-Spam-Status: No, hits=0.0 required=5.0 tests=none autolearn=ham version=2.60-r
c3

This is a spoofed message.

Email Computer Forensics



Do You Do Any of These Embarrassing Things?

Flag for follow up.

T

tschwarz@calprov.org
Fri 11/12/2021 4:55 AM
To: tschwarz@calprov.org

I am sorry to inform you but your device was hacked.

That's what happened. I have used a Zero Click vulnerability with a special code to hack your device through a website.
A complicated software that requires precise skills that I posess.
This exploit works in a chain with a specially crafted unique code and such type of an attack goes undetected.
You only had to visit a website to be infected, and unfortunately for you it's that simple for me.

You were not targeted, but just became one of the many unlucky people who got hacked through that webpage.
All of this happened in August. So I've had enough time to collect the information.

I think you already know what is going to happen next.
For a couple of month my software was quietly collecting information about your habits, websites you visit, websearches, texts you send.
There is more to it, but I have listed just a few reasons for you to understand how serious this is.

Email Computer Forensics

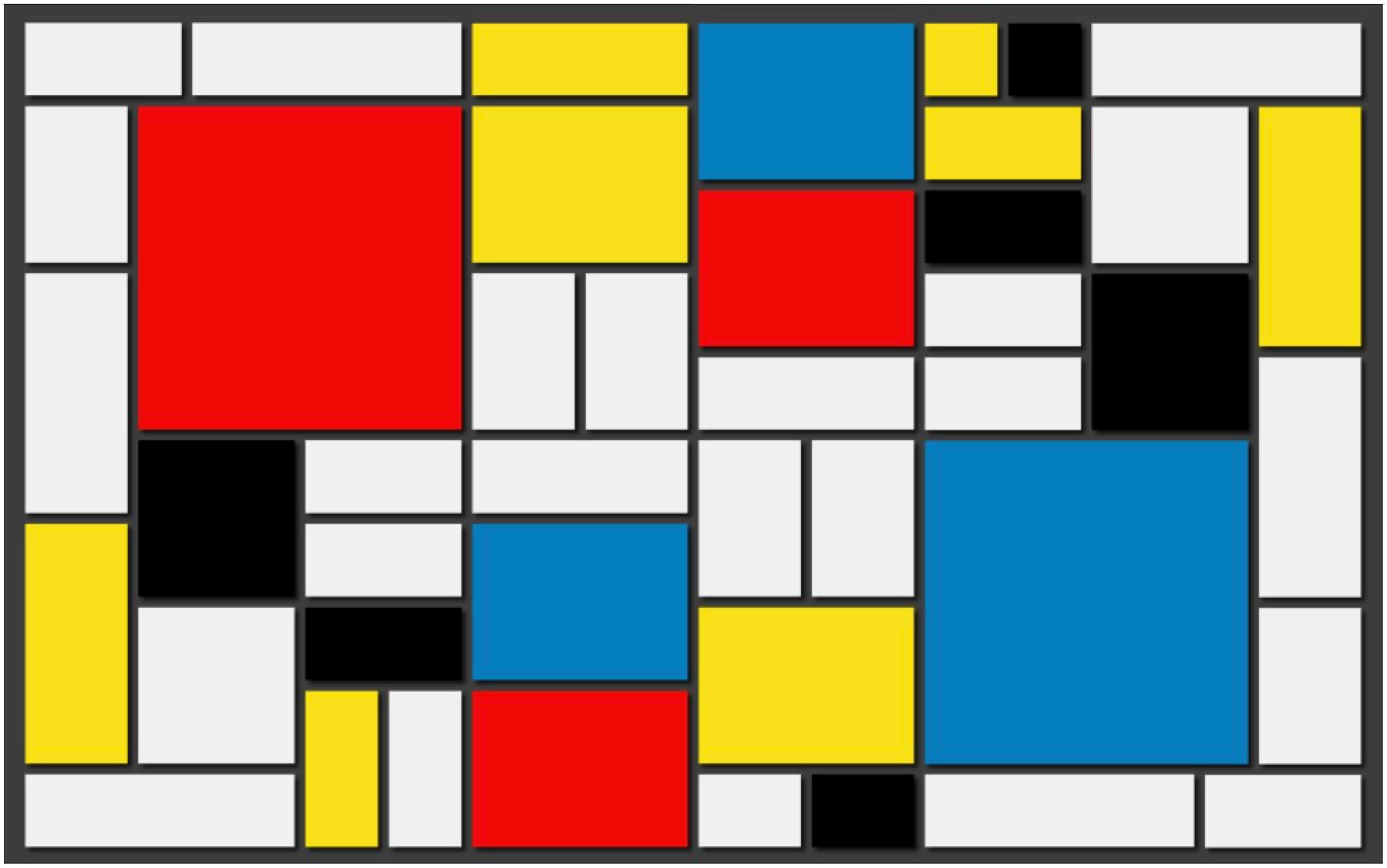
- Headers are added on the top
  - So we read them bottom to top
  - Very early on we might find some clues

```
(2603:10b6:802:21::18) with Microsoft SMTP Server (version=TLS1_2,
 cipher=TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384) id 15.20.4669.11 via Frontend
 Transport; Fri, 12 Nov 2021 10:55:33 +0000
Authentication-Results-Original: spf=fail (sender IP is 46.96.183.131)
 smtp.mailfrom=calprov.org; calprov.org; dkim=none (message not signed)
 header.d=none;calprov.org; dmarc=permerror action=none
 header.from=calprov.org;
Received-SPF: Fail (protection.outlook.com: domain of calprov.org does not
 designate 46.96.183.131 as permitted sender) receiver=protection.outlook.com;
 client-ip=46.96.183.131; helo=[46.96.183.131];
```

Email Computer Forensics

- We occasionally can glance some data
  - Though low-level bandits are more likely to use web-based emails
  - And high-level bandits use a botnet

```
whois 46.96.183.131

role:           Limited Liability Company "lifecell" NCC
org:            ORG-LLC4-RIPE
remarks:        LLC "lifecell" - ADMINISTRATIVE/TECHNICAL CONTACT
address:        11a, Solomyanska str, Kyiv, Ukraine, 03110
nic-hdl:        LFC-RIPE
abuse-mailbox:  ripe@lifecell.com.ua
mnt-by:         LIFECELL-MNT
created:        2016-09-27T13:07:53Z
last-modified:  2021-06-23T12:06:11Z
source:         RIPE # Filtered
admin-c:        VOYT-RIPE
admin-c:        OO1595-RIPE
tech-c:         VOYT-RIPE
tech-c:         OO1595-RIPE
```



ASN

ASN.1

- Standardized in 1984
  - by International Telegraph and Telephone Consultative Committee,
  - now called ITU-T,International Telecommunication Union - Telecommunication Standardization Sector)
- X.409 Recommendation
- Adopted by ISO

ASN.1

- Used to define data send over networks
- Standard that defines compound data structures
  - Basic Components:
    - integers, booleans, character strings, bit strings, …
  - Compound
    - structures, lists, choice, …

ASN.1

- Basic Encoding Rules
  - Takes an ASN.1 data type and turns it into a sequence of bits
  - Uses Tag-Length-Value encoding

ASN.1

- Example: C++ structure

```
struct GetRequest {
    int headerOnly; // flag: do we only want headers?
    int lock; // flag: should we checkout the object?
    string url; // the url to fetch
    AcceptTypes* acceptTypes;
    // Optional list of accept types that only apply
    // to this request
};
```

ASN.1

- Example: C++ structure

```
struct AcceptTypes {
    List<bitset>* standardTypes;
    // list of bitmaps indicating which
    // preference order for standard types
    List<string>* otherTypes;
    // nonstandard types in preference order
};
```

ASN.1

- Example: ASN.1

```
GetRequest ::= [APPLICATION 0] IMPLICIT SEQUENCE {
    headerOnly BOOLEAN,
    lock BOOLEAN,
    acceptTypes AcceptTypes OPTIONAL
    url OCTET STRING,
}


AcceptTypes ::= [APPLICATION 1] IMPLICIT SEQUENCE {
    standardTypes [0] IMPLICIT SEQUENCE OF BIT STRING
    {html(0),plain-text(1),gif(2), jpeg(3)} OPTIONAL,

    otherTypes [1] IMPLICIT SEQUENCE OF OCTET STRING OPTIONAL
}
```

ASN.1

- Example Data in ASN.1

```
(GetRequest
   :headerOnly TRUE
   :lock FALSE
   :acceptTypes (AcceptTypes
      :standardTypes ((html) (plain-text)))
   :url "/cv/tschwarz/mscsnet.mu.edu"
)
```

ASN.1

- BER:
  - Tags: single byte
    - Leading two bits determine class
      - Official ISO, Application-wide, Private, or only for this structure
  - Third bit: simple or compound
  - Remaining five bits
    - Tag number
    - If all ones: next byte contains a larger tag number

ASN.1

- BER
  - Definite Length
    - Up to 127: a byte starting with 0-bit
    - ≥128: a byte starting with 1-bit followed by as many bytes as indicated by the all but first bit
  - Indefinite Length
    - Zero byte, followed by length followed by two zero bytes

ASN.1

```
0x60 -- [0110|0000], [APPLICATION 0, Compound] - GetRequest
0x80 -- [1000|0000], Indefinite length

0x01 -- [0000|0001], [BOOLEAN] GetRequest.headerOnly
0x01 -- [0000|0001], length 1
0x01 -- [0000|0001], value TRUE

0x01 -- [0000|0001], [BOOLEAN] GetRequest.lock
0x01 -- [0000|0001], length 1
0x00 -- [0000|0000] value FALSE

0x61 -- [0110|0001], [APPLICATION 1, Compound] - GetRequest.types
0x80 -- indefinite length

0xa0 -- [1010|0000], [CONTEXT 0, Compound] types.standardTypes
0x80 -- indefinite length

0x03 -- [0000|0011] [BIT STRING]
0x02 -- length 2
0x04 -- Four bits unused
0x80 -- [1000|0000] {html}

0x03 -- [0000|0011] [BIT STRING]
0x02 -- length 2
0x04 -- Four bits unused
0x40 -- [0100|0000] {plain-text}

0x00
0x00 -- End of list of standard Types
0x00
0x00 -- End of Accept Types

0x04 -- [0000|0100], [OCTET STRING] GetRequest.url
0x16 -- [0001|1011], length 27
[/cv/tschwarz/mscsnet.mu.edu] -- value

0x00
0x00 -- End of get request
```

ASN.1

- PER
  - Tags only generated when needed to preserve ambiguity
  - Information not necessarily aligned with byte boundaries
  - Much more compact

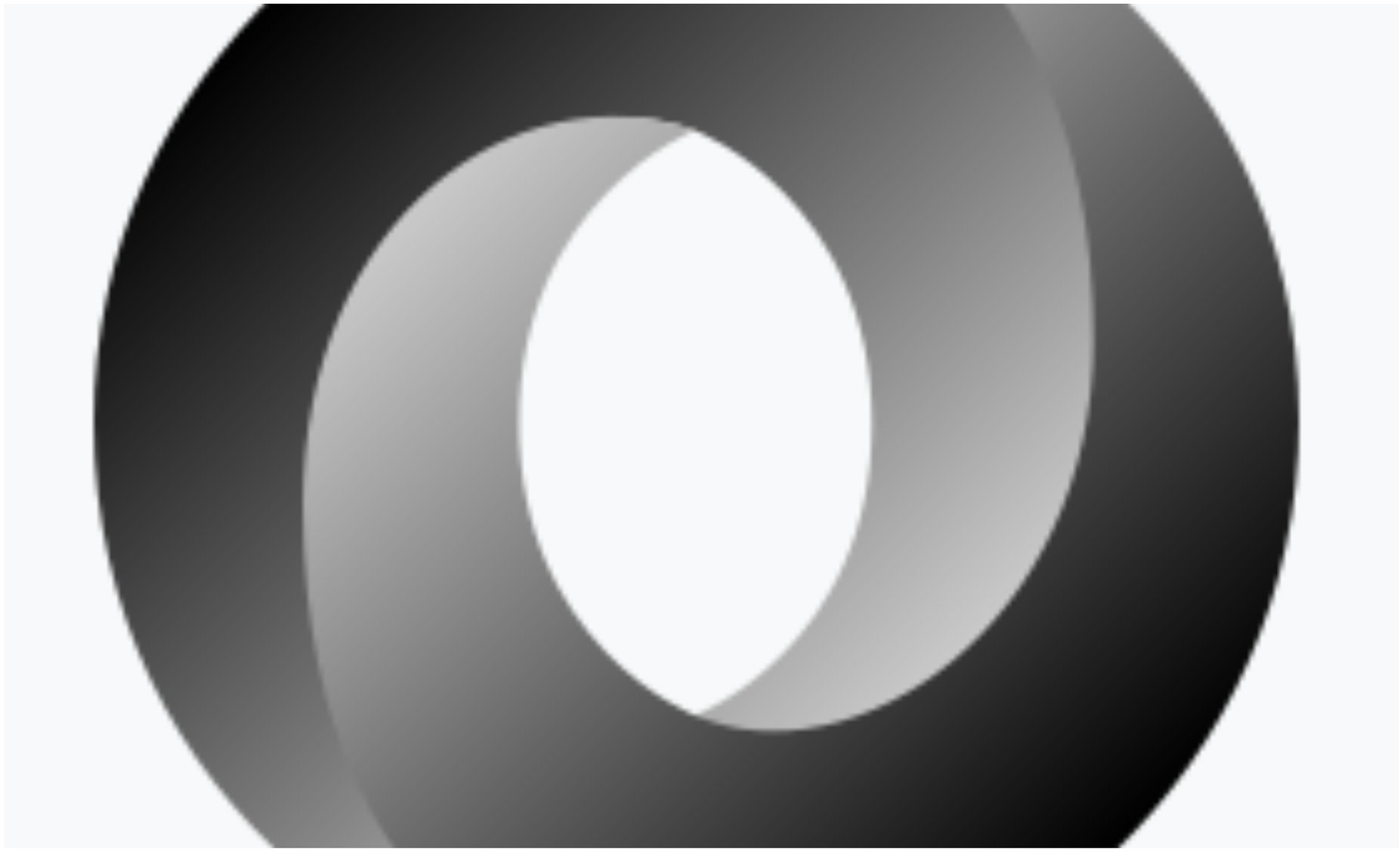
ASN.1

- PER Example

```
[1] -- flag bit indicates acceptTypes is present
[1] -- boolean, header only, TRUE
[0] -- boolean, lock, FALSE
[1] -- flag bit, indicates standardTypes is present
[0] -- flag bit, indicates otherTypes is absent
[000] -- pad bits to make length octet aligned
[00000010] -- length of 2, -- 2 standardType bit strings to fo
[1000] -- the first bit string, html is set
[0100] -- the second bit string, plain-text is set
[00011010] -- length 27; url is 22 octets long
/cv/tschwarz/mscsnet.mu.edu -- value of url
```

ASN.1

- Simple Data Types:
  - Integer32
  - Unsigned32
  - Octet String
  - Object Identifier



JSON

JSON

- Java Script Object Notation
  - Completely language independent
  - Based on Javascript object syntax
- Similar to XML
  - Plain text
  - Human readable and self explaining
  - Hierarchical
- But
  - Allows data types (XML: strings only)
  - No namespaces, no validation

JSON

- JSON:
  - Unordered: Comma separated list of key-value pairs enclosed in braces { }

```
var facultyData = {
    "MU ID": 123454321,
    "name": "Thomas Johannes Emil Schwarz, SJ",
    "department": "Computer Science",
    "degree": "Dr rer nat, Math"
  };
```

JSON

- Ordered:
  - Embraced by brackets: [ ]

JSON

- Data is sent as a string
  - Usually UTF-8 encoded for maximum compatibility
- AJAX (Asynchronous JavaScript And XML) applications now use more JSON than XML
- Most languages appropriate for building web sites (Python, PHP, Javascript) interact with JSON

Authentication Primer

Thomas Schwarz, SJ

Cryptography

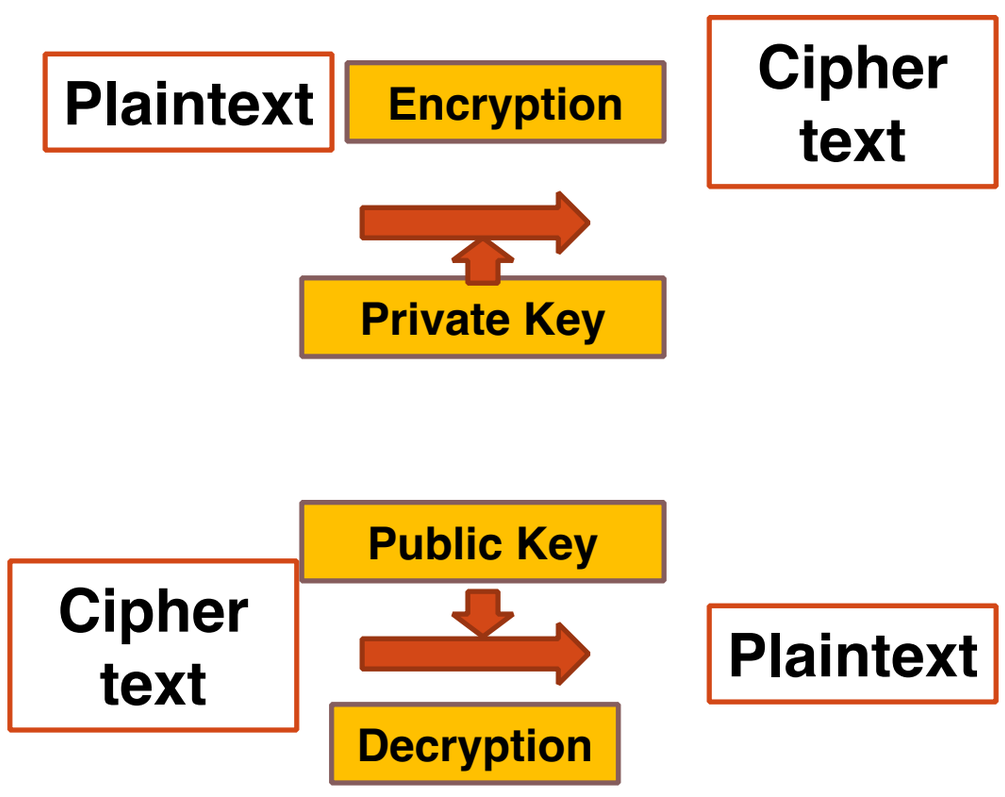
- Traditional cryptography uses a shared secret
  - Silly Example: Caesar Cipher
    - Take a message
    - Shift all characters circularly by a shift value
      - E.g. shift of 13:
        - A -> N, B -> O, C -> P, …, M -> Z, N -> A, Y -> L, Z -> M
      - ‘VENIVIDIVICI’ -> ‘IRAVIVQVIVPV’
- Secret is the shift value

Cryptography

- Given the shift value, we can now decode by shifting back

Cryptography

- Asymmetric cryptography uses <u>different keys</u> for encryption and decryption



- This is counter-intuitive, so we do the mathematics

Cryptography

- RSA algorithm:
  - Choose a very large number $n = pq$, which is the product of two very large prime numbers
  - Find (with number theory from p and q) two numbers $e, d \in \{2, \ldots, n-1\}$ such that always $\left(x^e\right)^d = x \pmod{n}$
    - Without knowing the decomposition $n = pq$ it is computationally impossible to calculate d from e and n

Cryptography

- RSA algorithm for authentication:
  - Make d the private key, but publish (e, n)
  - Convert a message into a (series of) number(s) between 0 and n.
  - Take the d-th power of the message
  - Publish the message and the d-th power of the message
  - I can now verify that the combined message comes from someone who knows d

Cryptography

- In reality:
  - Calculating large exponents takes time
    - Even though you can do so smartly
      - E.g. $2^{100} = 2^{64} \times 2^{32} \times 2^4$
        - And calculate $2^2 = 4$, $2^4 = 4^2 = 16$, $2^8 = 16^2 = 256$, $2^{16} = 256^2 = 65536$
- with costs $O(\log(e))$

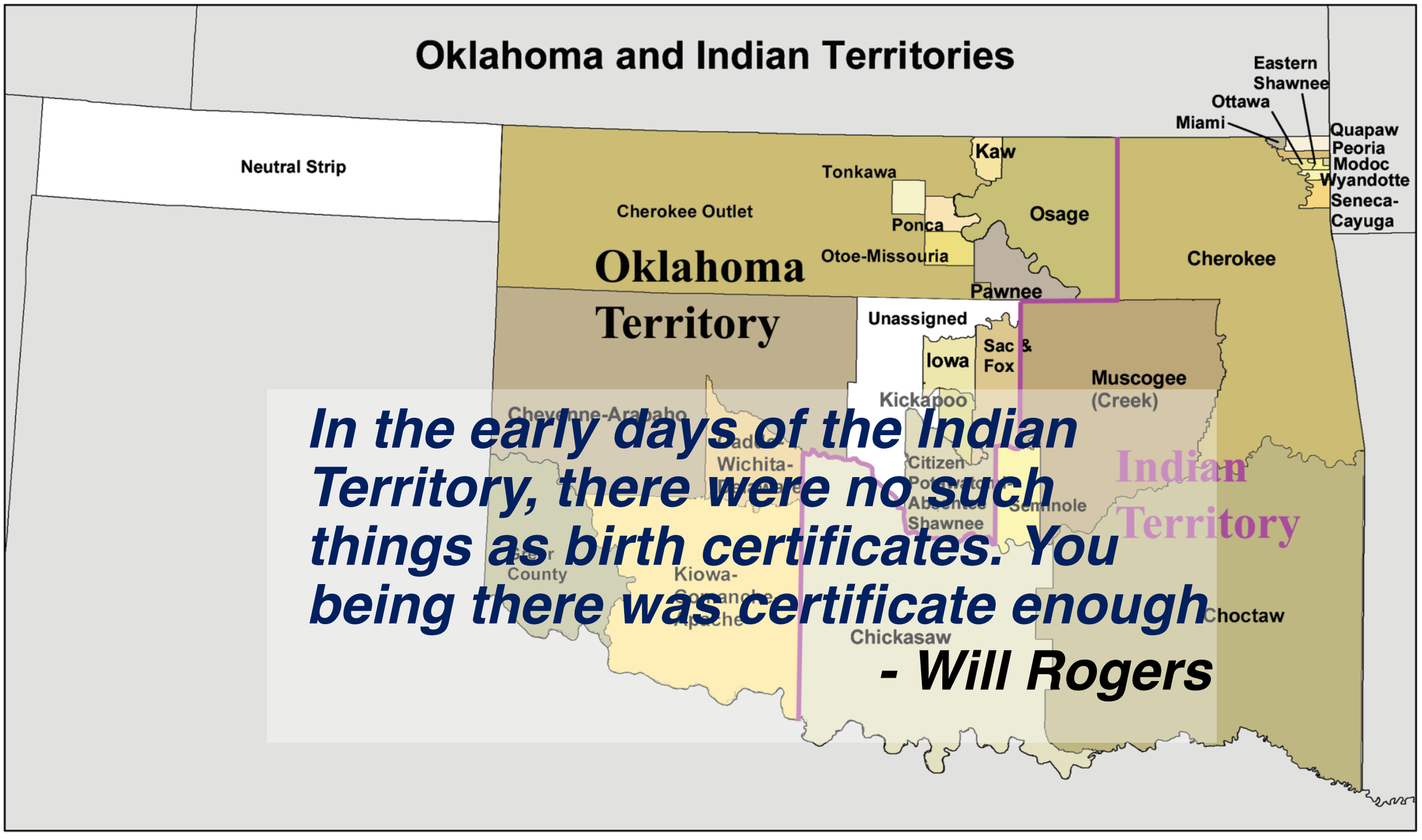
Cryptography

- In reality:
  - Use a “cryptographically secure” hash function h
  - Sign message m by providing
    - $\left(m \quad , \qquad h(m)^d \pmod{n}\right)$
- To authenticate, I decipher $\left(h(m)^d\right)^e$, which should be $h(m)$ and recalculate the hash myself
- If everything works out, I have authenticated

Authentication Infrastructure

- In order to trust that a given “certificate” (n, e) comes from the correct party
- Use a Public Key Infrastructure : PKI

Public Key Infrastructure


Oklahoma and Indian Territories
Neutral Strip
Cherokee Outlet
Oklahoma Territory
Tonkawa
Kaw
Ponca
Osage
Otoe-Missouria
Pawnee
Unassigned
Iowa
Sac & Fox
Kickapoo
Cheyenne-Arapaho
Wichita-
Citizen
Absentee Shawnee
Seminole
Kiowa-
Chickasaw
Choctaw
Muscogee (Creek)
Cherokee
Eastern Shawnee
Ottawa
Miami
Quapaw
Peoria
Modoc
Wyandotte
Seneca-Cayuga
Indian Territory
In the early days of the Indian Territory, there were no such things as birth certificates. You being there was certificate enough
- Will Rogers

Certificates

- Certificates link entity's name to its public key
- Are signed by a certifying authority

```
Alice has public key 145029348579 7signed Carol
```

Alice is the *subject*
Carol is the *issuer*
If Bob uses the certificate,
he becomes the *verifier*

Certificates

- Need to specify algorithms used
- Add validity dates

X.509

X.509 Version Number

Serial Number

Signature Algorithm Identifier

Issuer (X.500 Name)

Validity Period (Start – Expiration dates / times)

Subject (X.500 Name)

Subject Public Key Information: Algorithm Identifier, Public Key Value

Issuer Unique Identifier

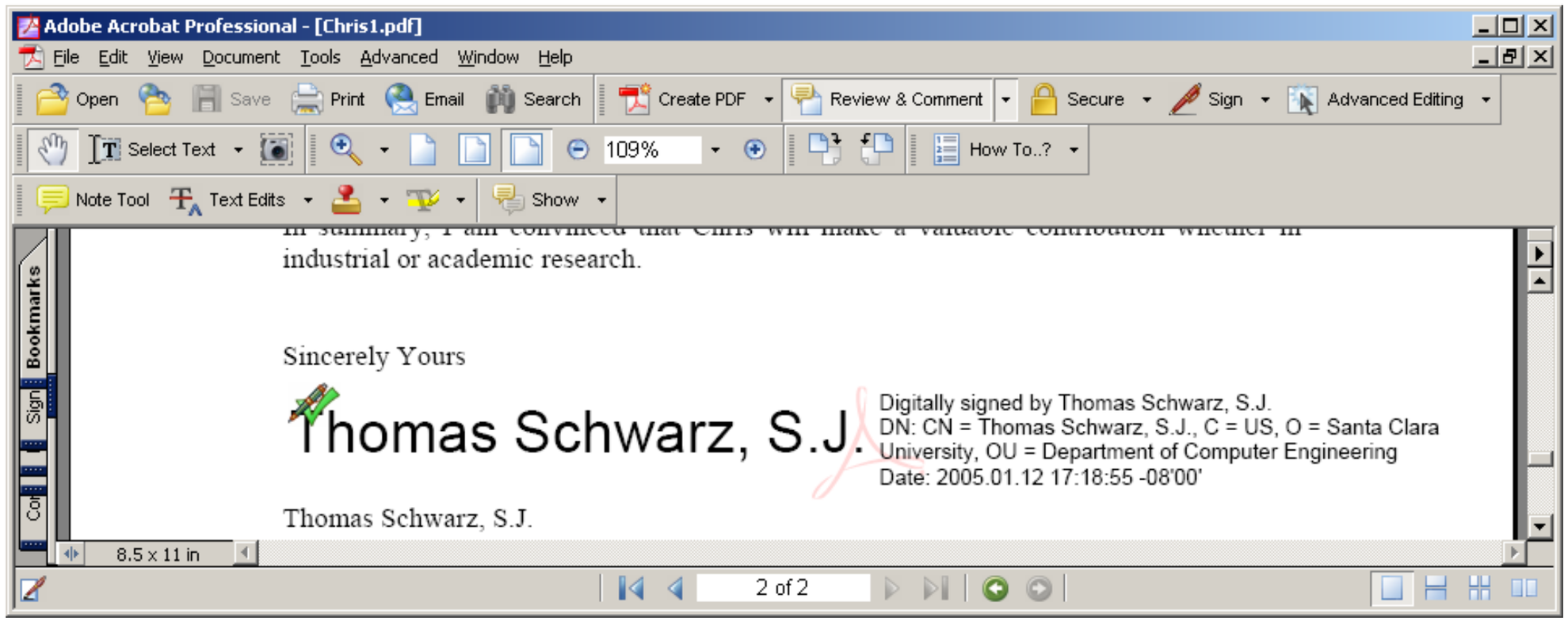
CA Digital Signature

Certificates

- Naming is a security concern
  - Example: “Microsoft Corporation” vs “Microsoft Corporation”
    - Look the same, but are different unicode strings (the second one uses the Hungarian letter o)
- How many “Bill Smith” are there?

Names

- X 500 names
  - Common name, country, organization, organizational unit



X.509 Certificates

- Uses X.500 identifiers for used algorithms
  - Issuer signature
  - Public key certified
  - Algorithms are registered and numbered in the Abstract Syntax Notation 1

X.509 Certificate

- VERSION — 2 or 3
- SERIALNUMBER — identifies uniquely certificates of issuer
- SIGNATURE — Method used for signing
- VALIDITY — start and end time of validity
- SUBJECT
- SUBJECTPUBLICKEYINFO
- ISSUERUNIQUEIDENTIFIER
- ALGORITHMIDENTIFIER
- ENCRYPTED

Value of Certificates

- Value of certificate depends on the diligence of the issuer in verification of the identity of the subject
  - Verisign once issued a certificate for Microsoft on a stolen credit card
    - Claimed to have changed procedures so that it could never happen again
    - But would not say how for security reasons

certmgr - [Certificates - Current User\Untrusted Certificates\Certificates]

File Action View Help

- Certificates - Current User
  - Personal
  - Trusted Root Certification Au
  - Enterprise Trust
  - Intermediate Certification Au
    - Certificate Revocation Lis
    - Certificates
  - Active Directory User Object
  - Trusted Publishers
  - Untrusted Certificates
    - Certificates
  - Third-Party Root Certificatior
  - Trusted People
  - Smart Card Trusted Roots

Issued To	Issued By	Expiration Date	Intended Purposes	Friendly Name	Status	Certificate Te...
Microsoft Corporation	VeriSign Commercial Software Pu...	1/31/2002	<All>	Fraudulent, NOT M...		
Microsoft Corporation	VeriSign Commercial Software Pu...	1/30/2002	<All>	Fraudulent, NOT M...		

Untrusted Certificates store contains 2 certificates.

Value of Certificate

- Need public / private key pairs and hence certificates for each use of asymmetric cryptography
  - One for signing
  - One for each authentication protocol
  - One for confidentiality

X.509 Extensions

- Version 3 allows extensions
  - Standard extensions
    - Key information
    - Policy information
    - Subject and issuer attributes
    - Restrictions on certificate path
    - Extensions for certificate revocation

PKIX

- Another standard from IETF 1994
  - Has extensions
    - AuthorityKeyIdentifier
    - SubjectKeyIdentifier
    - KeyUsage
    - PrivateKeyUsagePeriod
    - CertificatePolicies
    - PolicyMappings
    - SubjectAltName

Other standards

- PBP
- WAP WTL — replaces ASN.1 names with simpler ones
- DNSSEC — certificates for DNS
- SPKI (Simple PKI) RFC 2693

Revocation

- Certificates need to be revoked because of
  - Issuer mistakes
  - Loss of private keys

Revocation

- Certificate Revocation Lists (CRL)
  - Basic Idea: publish a list of revoked certificates periodically
  - Certificates are identified by serial number and issuer
  - Security problems:
    - Confidentiality:
      - Publish only serial number and hash of contents
    - Adversary can publish old CRL
      - Always publish a CRL at a given time
      - Sign CRL

Revocation

- Need to save space:
  - Delta CRL
    - Publish difference between previous and current CRL
  - Include first valid serial number
    - Allows to revoke old certificates en masse

Implementation of Revocation

- On-Line Revocation Service (OLRS)
  - Server that responds whether a certificate is valid
- Black List versus White List
  - Black List of revoked certificates
  - White List of valid certificates
  - Both prevent falsifying a certificate with a used serial number

PKI modes

- Certificate chains
  - Verifier needs to know public key of issuer
  - Might need a certificate for issuer
  - Gives raise to chains of certificates

PKI models

- Who can become an issuer?
  - Anarchy (PGP)
    - Everybody can sign certificates
    - Verifiers have a database of certificates
    - Verifier assigns trust to certificates
      - Usually multiple chains for the same subject — public key pair

PKI models

- Monopoly
  - Only one entity (certifying authority) in the universe
    - Its public key is embedded in all software and hardware products
- Who should this entity be?

PKI models

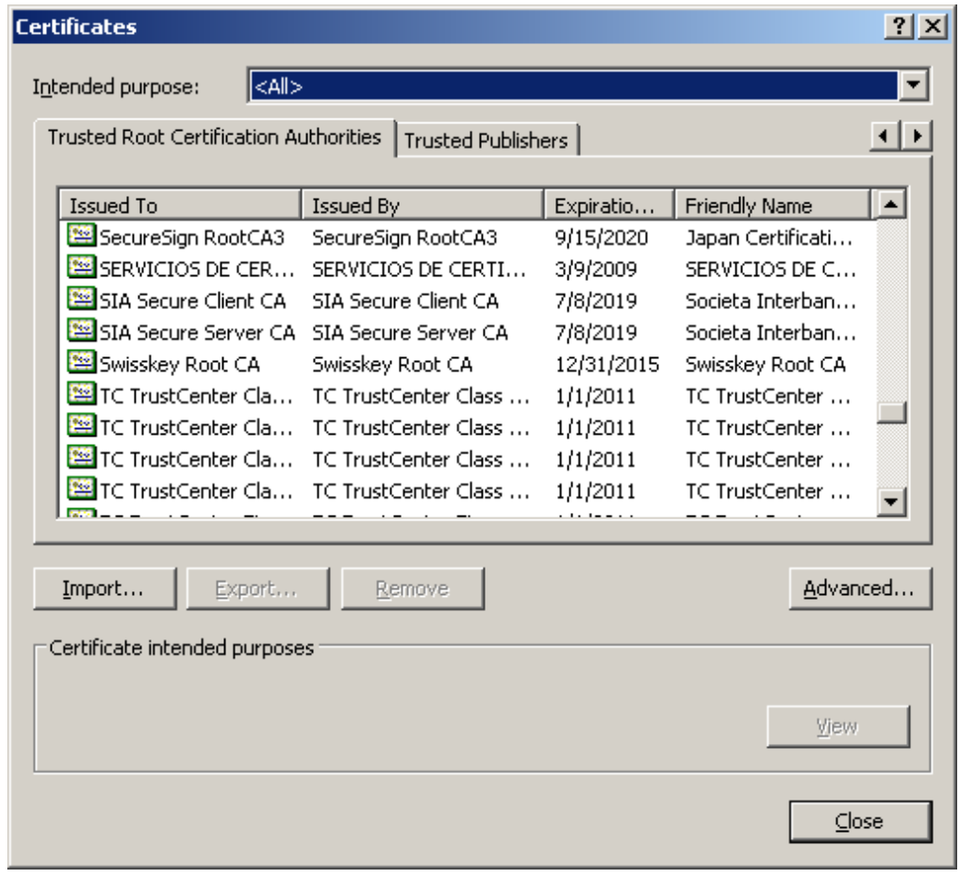
- Monopoly with Registration Authorities (RA)
  - Sole CA allows RA
    - RA authenticate the identity of a subject, create keys, and guarantee the link between subject and key
  - All certificates are from the same CA

PKI models

- Monopoly with delegated CA
  - Root CA authenticates other CAs
  - CAs sign their own certificates

PKI models

- Oligopoly
  - There are several trusted CAs
    - Software / OS manufacturer decide which CAs are trusted

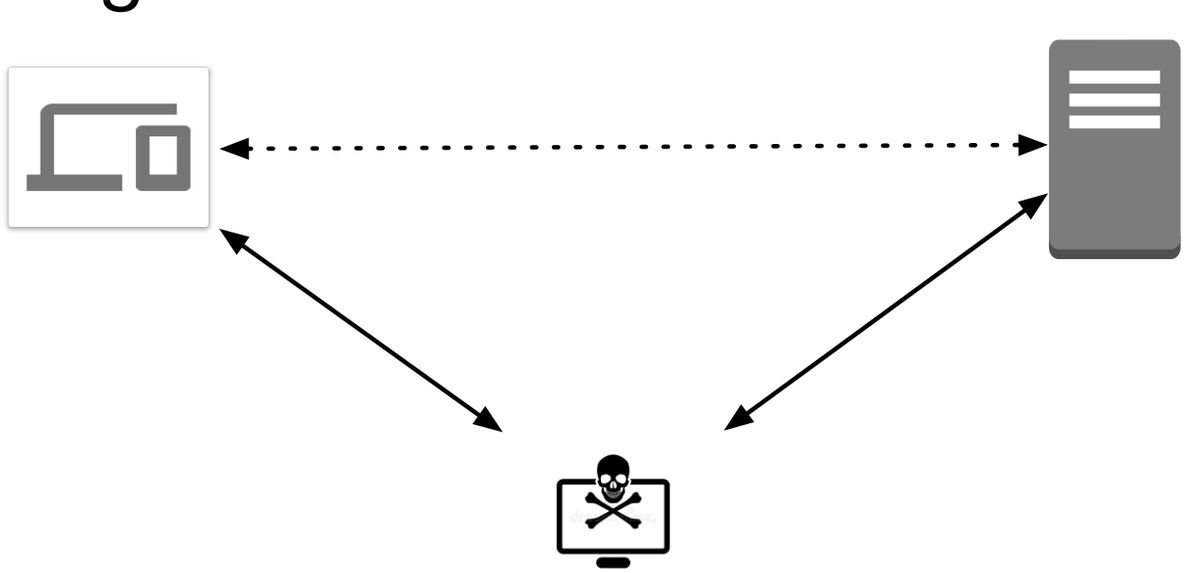


PKI models

- Domain CAs
  - Certificates for users in a certain domain
    - Each CA administers its own domain
    - Possible to allow cross-site certification
    - PKIX allows to restrict certificates to certain domains

HTTPS

- End-to-end secure protocol for web
  - Designed to be secure against network attackers
    - Including Man-in-the-Middle attack:



- Traffic is directed through the adversary’s machine

HTTPS

- Provides
  - encryption
  - authentication of server
  - integrity checking

HTTPS

- HTTPS goals:
  - Identify secure connections
    - Uses SSL/TLS (see next class)
    - Lock icon identifies pages secure against network attacks
      - Browser manufacturers have significantly improved user interface design and semantics
- HTTPS guarantees:
  - The origin of the page is what it is in the address bar
  - Contents of the page have not been modified by a network attacker

HTTPS

- Mechanism relies on asymmetric cryptography
  - Guarantees that public key of web-site is owned by the web-site
  - Browsers install root **certificate authorities** (CAs)
    - Or defer to the OS (e.g. on Mac)
    - CAs are usually in the hundreds
    - A public key for any website is accepted if it is certified by one of these CAs

HTTPS

- Extended validation certificates:
  - Certificate request must be approved by a human lawyer at the certificate authority

HTTPS

- Certificate revocation:
  - Private key corresponding to the certified public key has been compromised
  - User stopped paying his certification fee to the CA and the CA no longer wishes to certify him
  - CA’s certificate has been compromised!

Security in the Networking Stack

- Higher layers do not have appropriate security functions
  - IPSEC is below TCP
  - Can only authenticate IP-addresses, but not users
  - as the application would like

General Security Concerns

- Session highjacking:
  - Authentication is done at the beginning of a session
  - Adversary can attack session maintenance mechanism such as a token (a cookie)
    - Steal (intercept) cookie of a valid session
    - Use cookie to give commands

General Security Concerns

- Session keys
  - Cryptanalysis might reveal a key if there are many interchanges
    - Therefore: use keys sparingly
  - Therefore: have both parties agree on a key for only a single session
  - This also protects against replays of previous messages

General Security Concerns

- Session keys need to be unpredictable
  - Goldberg & Wagner found that an early SSL implementation used machine state to generate random session key
    - Component of machine states are highly correlated
  - Goldberg & Wagner managed to reduce the space of session keys to something manageable

General Security Concerns

- Generation of session keys
  - Random number generation is difficult
    - Only now do we have hardware random number generation
  - Software technologies have become much better
  - Normal protocols:
  - Have both sides contribute to key generation
    - Session keys as random as the best of the two

General Security Concerns

- Packet numbering
  - To prevent reorder and replay attacks
    - Have all packages be numbered
  - Even if this means that packages have two serial numbers
    - E.g. One for SSL, one for TCP

General Security Concerns

- Perfect forward security
  - Prevent after-after-the-fact breaches of confidentiality
    - Use random session keys and throw them away afterwards
      - A sniffer can record an encrypted conversation
      - During the conversation, the two parties and possibly the sniffer has the key information
      - After the conversation, no-one has the key

General Security Concerns

- Escrow foliage
  - In many jurisdictions, keys used for encryption need to be put in escrow
  - With session keys, the state agency can no longer use an escrowed key to reconstruct session keys
  - Therefore, can only intercept communication after the escrowed key was obtained

General Security Concerns

- Protection against denial of service attacks
  - DoS attacks are easy if the attacker has less work to do than the victim
    - E.g. Victim deciphers random packages or victim creates and serves information for random requests

General Security Concerns

- Protection against denial of service attacks
    - Photoris cookies
    - When server Bob receives a request from address S
        - Bob calculates a cookie from S
        - Only accepts requests from S with the cookie
            - Key: Fast to calculate and verify cookie
        - Can throttle initial requests without introducing state

General Security Concerns

- Protection against denial of service attacks
  - Puzzles:
  - When Alice requests service from Bob
    - Bob gives her a puzzle to solve
    - Alice only receives service after she solved the puzzle
    - Bob is much faster at creating puzzles and checking solutions than is Alice at solving them
    - But Alice can do so fast enough to not make it visible to her human user

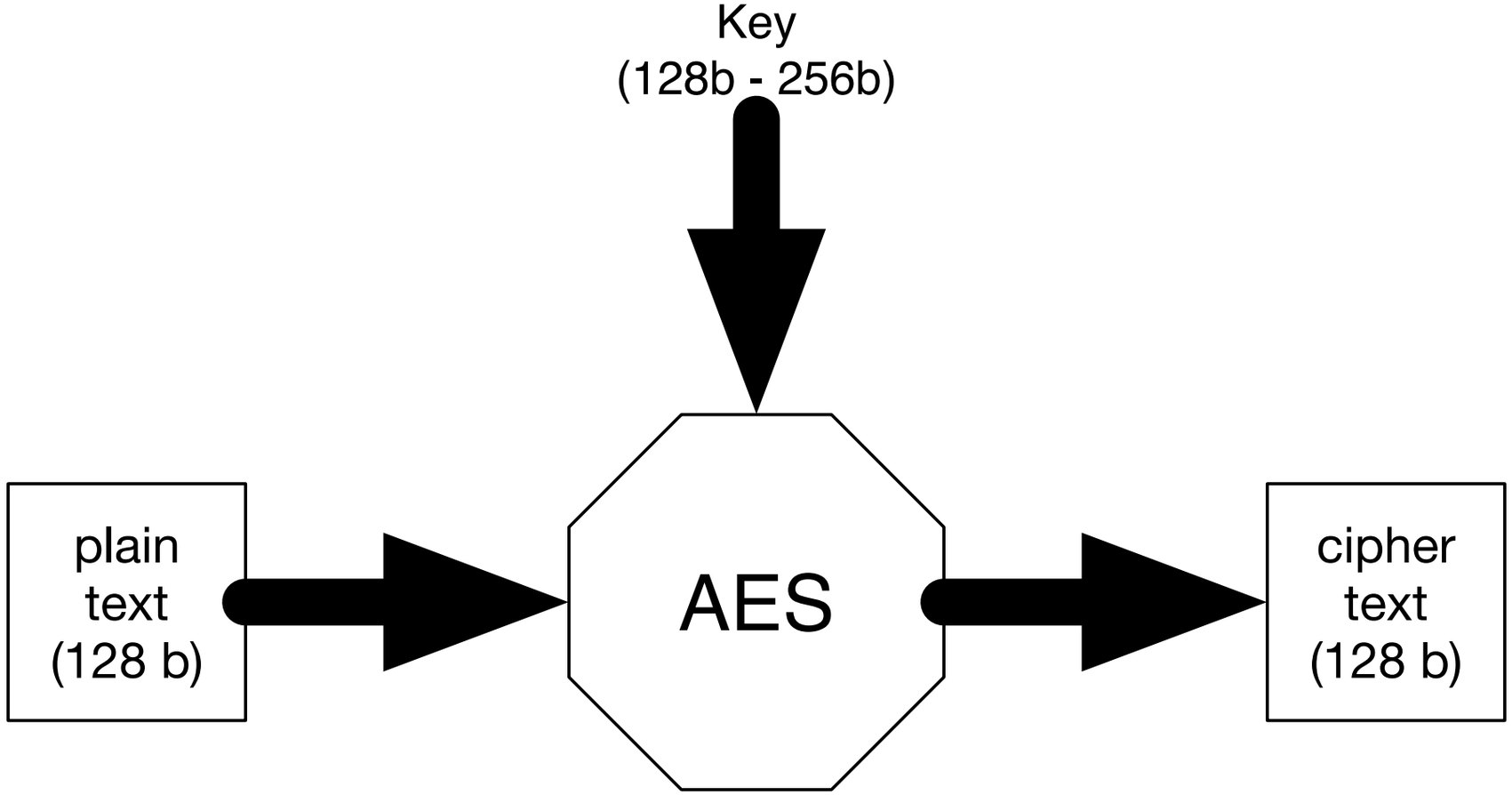
V. Rijmen

J. Daemen

AES

AES Encryption

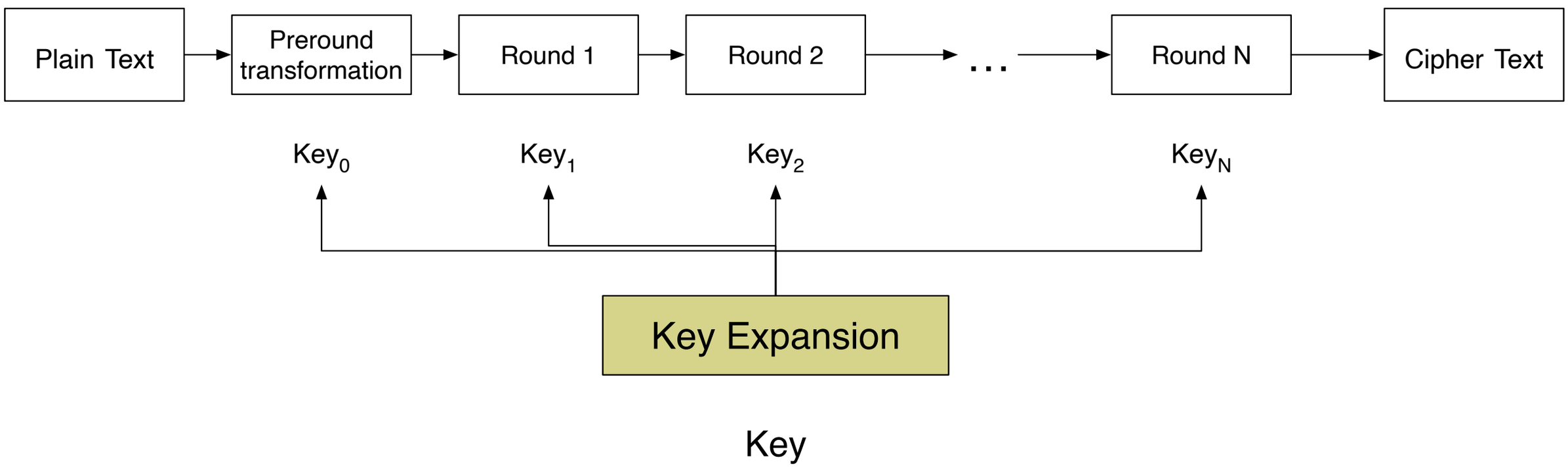
- Block-based cypher:



AES Encryption

- Each block is encrypted in rounds
  - Key length 128b: 10 rounds
  - Key length 192b: 12 rounds
  - Key length 256b: 14 rounds
- For each round, a different key is produced from the initial key

AES Encryption

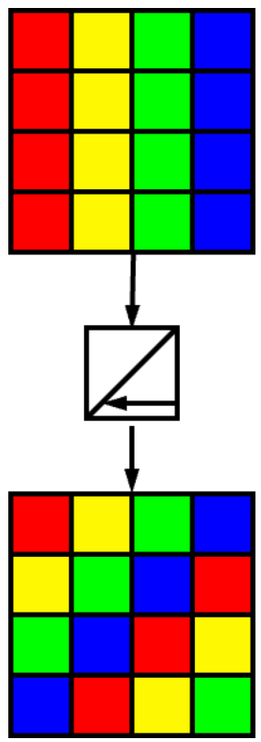


AES Encryption

- Write block as a 4×4 block of bytes
- Each round:
  - Replaces each byte with a byte using a substitution table
  - S-box: A fixed permutation of 8 bits
  - Derived algebraically in $\mathscr{Z}_{256}$, a finite field

AES Encryption

- Write block as a 4×4 block of bytes
- Each round (based on round key):
  - Replaces each byte with a byte used a substitution table
  - Shifts rows cyclically



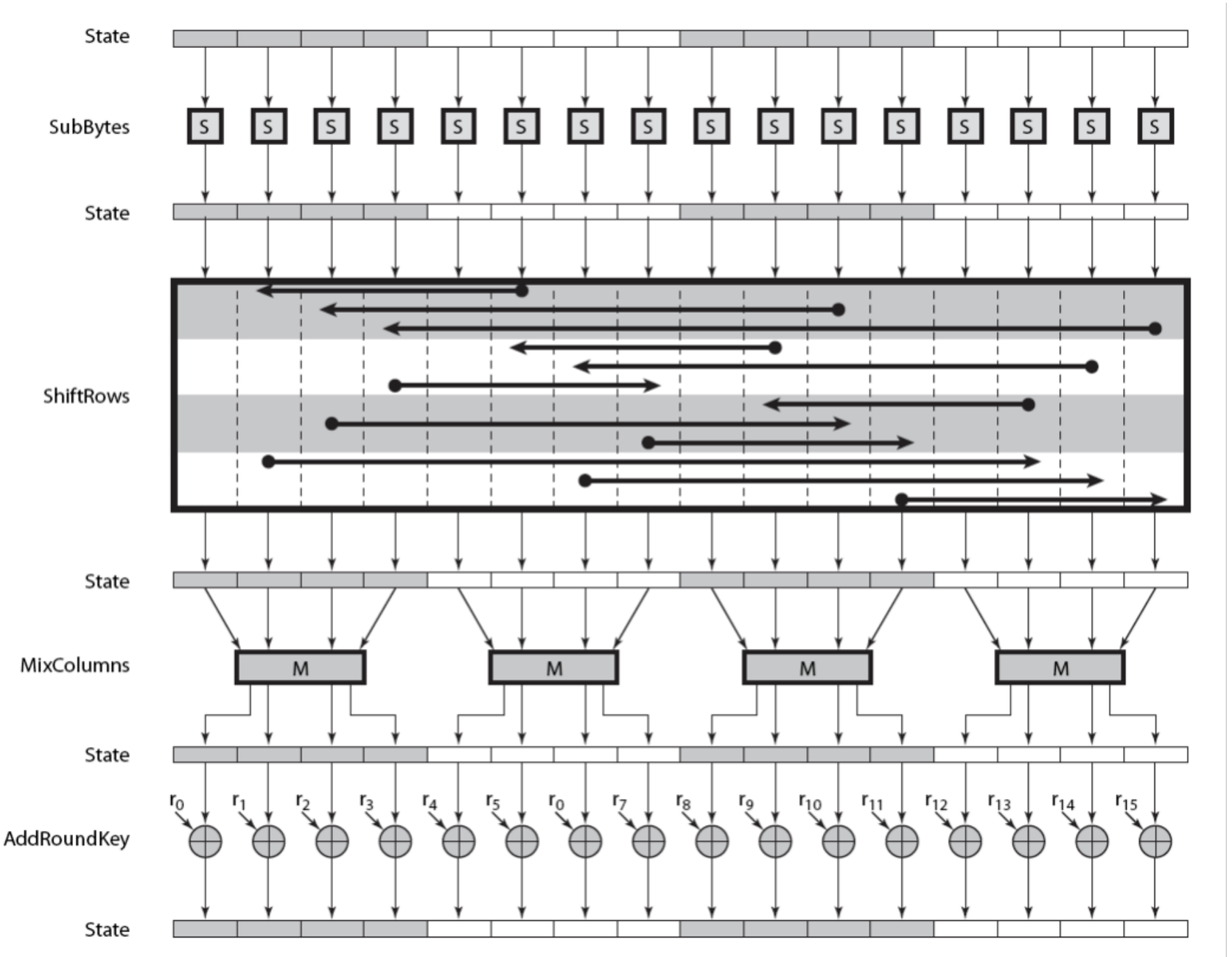
AES Encryption

- Write block as a 4×4 block of bytes
- Each round (based on round key):
  - Replaces each byte with a byte used a substitution table
  - Shifts rows
  - Transform each column by a linear transformation $C \to \mathbf{A} \cdot C$ with invertible matrix

AES Encryption

- Write block as a 4×4 block of bytes
- Each round (based on round key):
  - Replaces each byte with a byte used a substitution table
  - Shifts rows
  - Transform each column by a linear transformation $C \to \mathbf{A} \cdot C$ with invertible matrix
  - Replace block by exclusive or-ing with the round key

AES Encryption



AES Encryption

- Intel, AMD, and other chip manufacturer created additional instructions for AES calculations
  - Culminating in one instruction per AES round
  - This incidentally avoids timing attacks

Authentication

Authentication

- Two different problems
  - Human-Computer Authentication
  - Computer-Computer Authentication

Human Computer Authentication

- Based on:
  - Knowledge
    - Passwords, Ability to answer questions, …
  - Possession
    - Token cards, ATM card, Physical keys
  - Identity
    - Biometrics, IP-address

Human Computer Authentication

- Typical attack vectors
  - Online vs Offline attacks
    - Password cracking vs shoulder-surfing
    - Eaves-dropping
      - e.g. shoulder-surfing: Protect by using graphical passwords
    - Spoofing / impersonation
    - Theft of credentials / theft of databases

Human Computer Authentication

- Offline password guessing:
  - Passwords are stored as *user_name*, *hash(password)* pairs
- Dictionary attack
  - Generate passwords from a dictionary
- Protection:
  - System administrator runs dictionary attacks against user passwords and sends warnings
  - Insist on strong passwords
  - Use salts to increase password space:
    - *user_name, salt, hash(salt + password)*

Human Computer Authentication

- Offline password guessing
  - Invert hash with rainbow tables
  - Protection against rainbow tables
    - Make passwords longer
    - Or combine with salts

Human Computer Authentication

- Offline attacks
  - Obtain passwords from other breaches
    - as most users reuse passwords
  - Protection (?): Develop password requirements unique to the site
    - Users will no longer remember all passwords
    - Password recovery mechanism is usually weak point

Human Computer Authentication

- Eaves-dropping / snooping
  - Capture password as it is being entered
  - Hardware: Fake keyboards, fake connectors, sniff wireless traffic between keyboard and computer
  - Software: Keyloggers installed by Trojans / Phishing

Human Computer Authentication

- NIST SP 800-63B Guidelines for passwords`At least 8 characters unless generated randomly by organization
  - Passwords need to be vetted against
    - Passwords obtained from previous breach corpuses.
    - Dictionary words.
    - Repetitive or sequential characters (e.g. ‘aaaaaa’, ‘1234abcd’).
    - Context specific words, such as the name of the service, the username, and derivatives thereof.

Human Computer Authentication

- NIST SP 800-63B Guidelines for passwords
  - Use throttling for repeated login attempts
  - NO composition rules (at least one digit, …)
  - NO forced password changes
  - Give client option to see entered password clearly
  - May display entered letter for a short period

Human Computer Authentication

- NIST SP 800-63B Guidelines for passwords
  - Organization has to use approved encryption to store passwords
  - Organization has to use salts and an approved hash
  - Has to iterate hash at least 10000 times or used a keyed hash
  - Organization has to use secure communication channel

Human Computer Authentication

- Authentication tokens (a.k.a. security tokens)
  - Possession of the token (and usually knowledge of a pin) authenticate
  - Passive tokens: Do not interact with the authenticating site
  - Active tokens: Can be used for challenge-response schemes

Human Computer Authentication

- Passive tokens:
  - Show a number on a display
  - Number switches every minute
  - User enters number for authentication
  - Authenticating site checks the number
    - against a list
    - against an algorithm
    - usually allows token to be slightly out of sync

Human Computer Authentication

- Biometrics
  - Retinal scanner
  - Fingerprint reader
  - Face recognition
  - Iris scanner
  - Handprint readers
  - Voiceprints
  - Keystroke timing
  - Signatures

Authentication Protocols

- Threat model
  - Passive sniffing
    - Malicious Mallory can read messages between Alice and Bob
    - but does not change / suppress them
  - Spoofing
    - Mallory pretends to be Alice or Bob
- Breaking Crypto
- Man-in-the-Middle
- Replay attacks
- Reflection attacks (open several sessions)

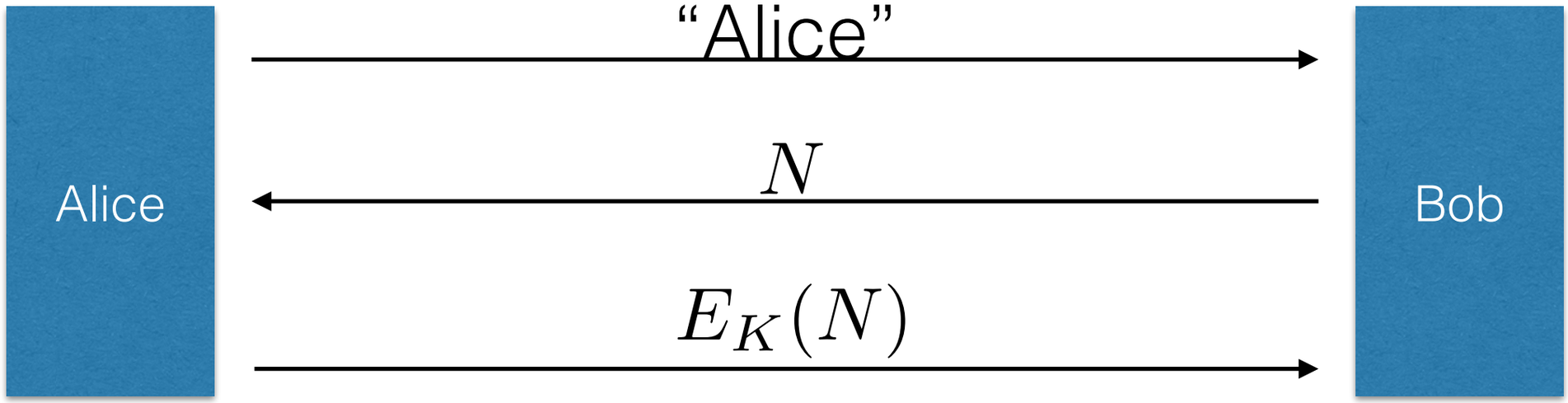
Authentication Protocols

- Simple password protocol

- Vulnerable to:
  - Sniffing
  - Spoofing (Mallory pretends to be Bob)
  - Replay attacks

Authentication Protocols

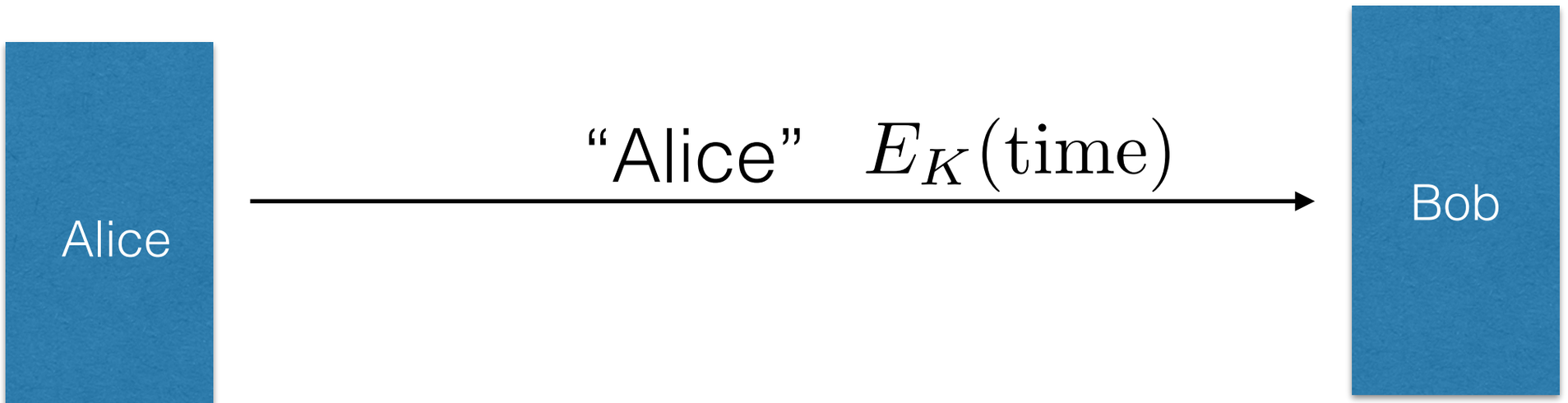
- One sided authentication (Alice to Bob)
  - Alice and Bob share secret K and use symmetric encryption



- Bob challenges Alice with a random number N.
  - A nonce
- Alice encrypts the random number
- Bob verifies the encryption

Authentication Protocols

- Clock-based scheme



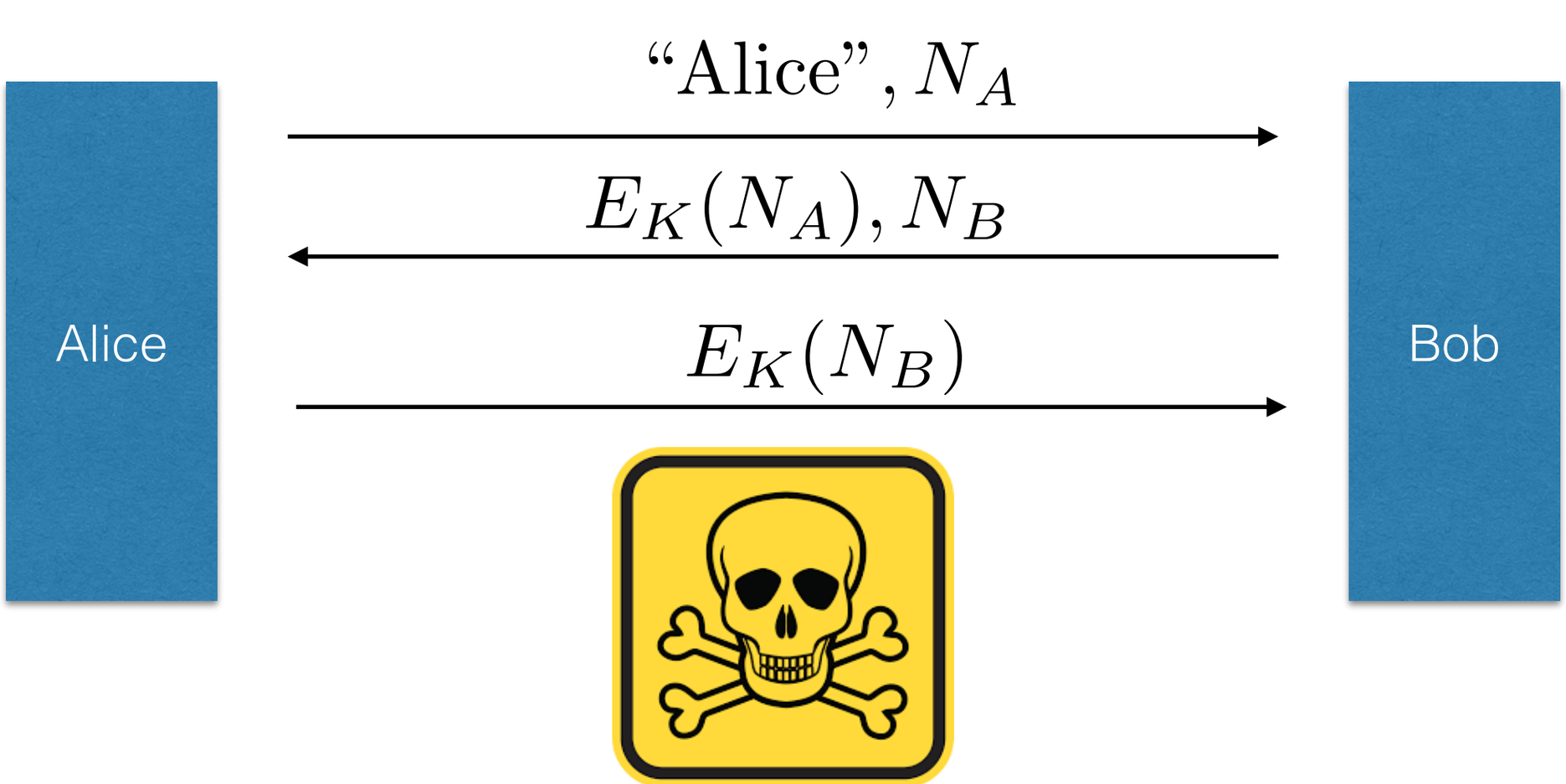
- Instead of a challenge, Alice just encrypts the time

Authentication

- Public key based one-sided authentication
  - Alice has private key *R* and public key *U*

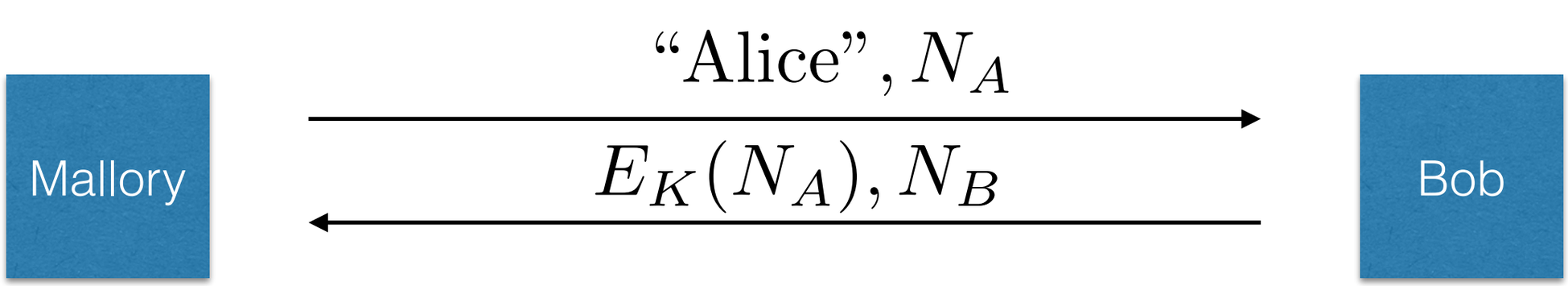
Mutual Authentication

- To save one round, one can have Alice challenge Bob first



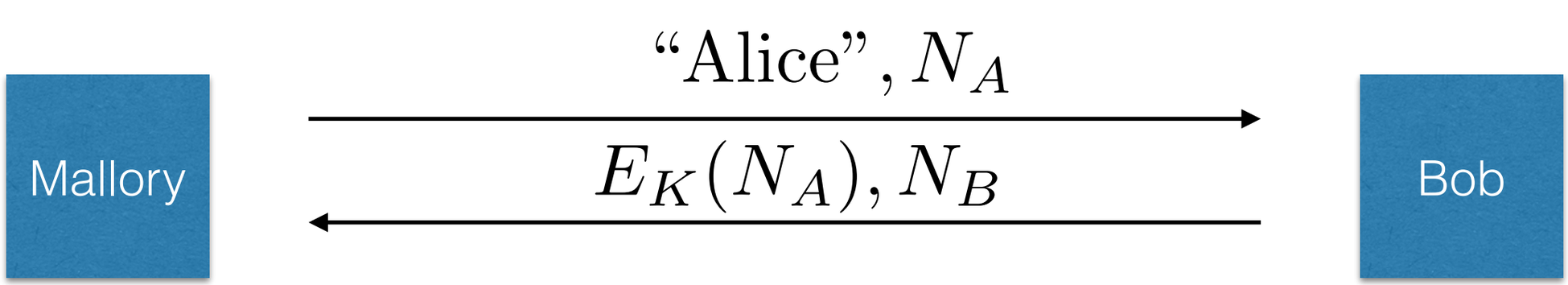
Mutual Authentication

- The three-round protocol is vulnerable to a replay attack
  - Mallory pretends to be Alice.



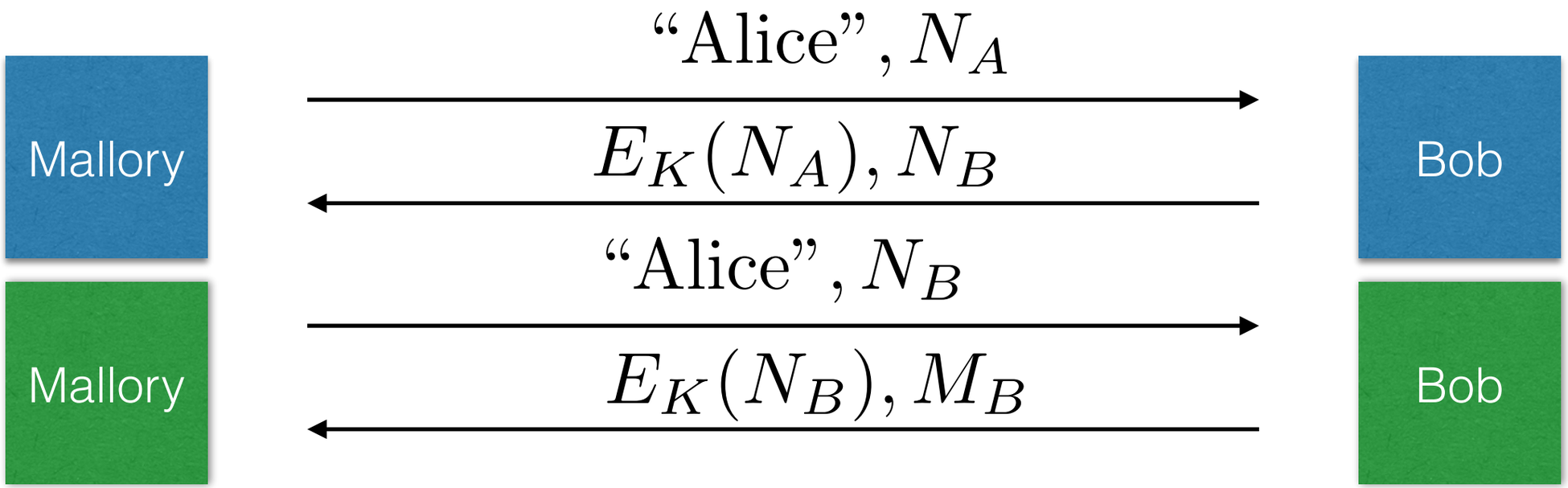
Mutual Authentication

- The three-round protocol is vulnerable to a replay attack
  - Mallory pretends to be Alice.



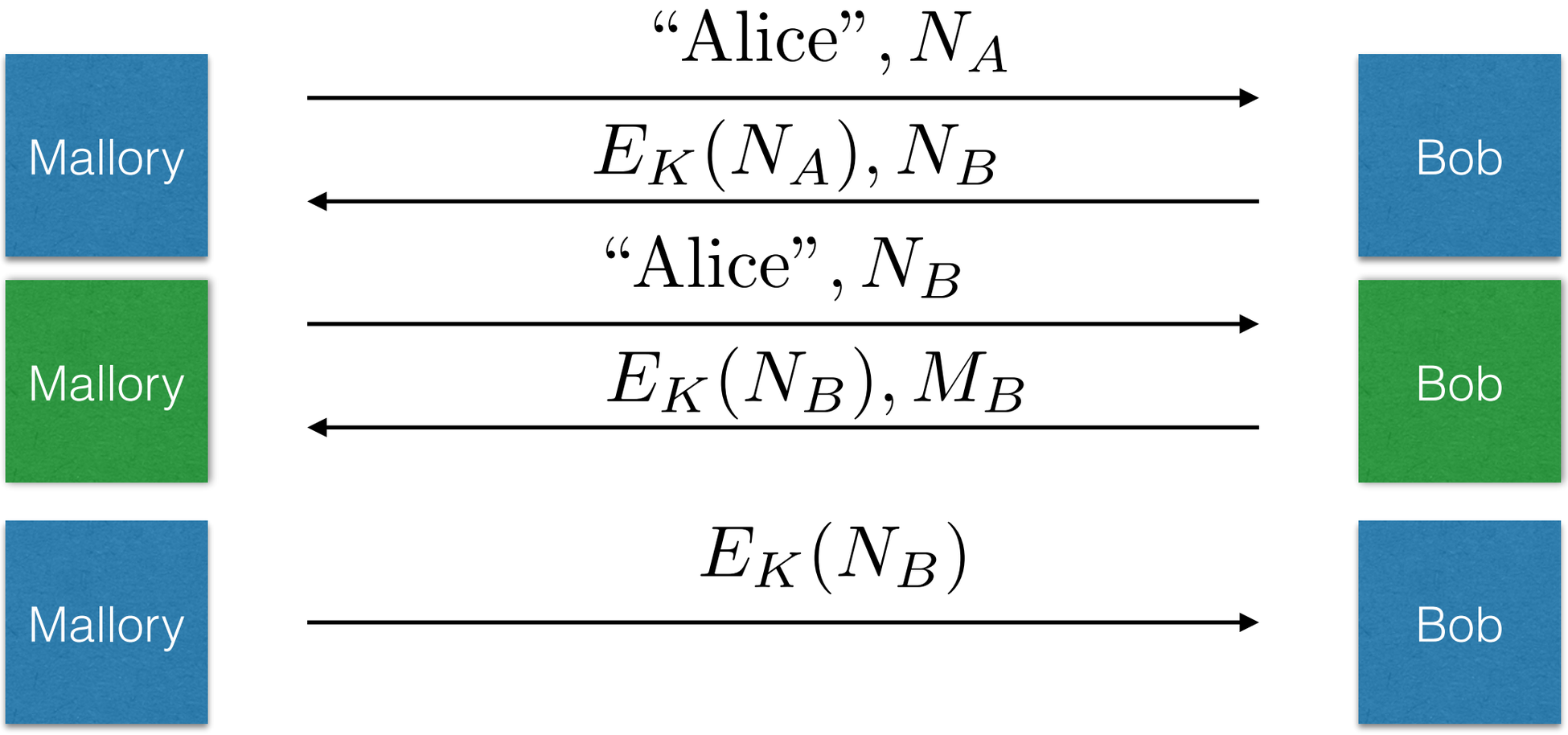
- Mallory is now stuck. (S)he opens a second connection to Bob.

Mutual Authentication



- Mallory reflects Bob's challenge back to Bob
- Bob solves the challenge and poses a new one

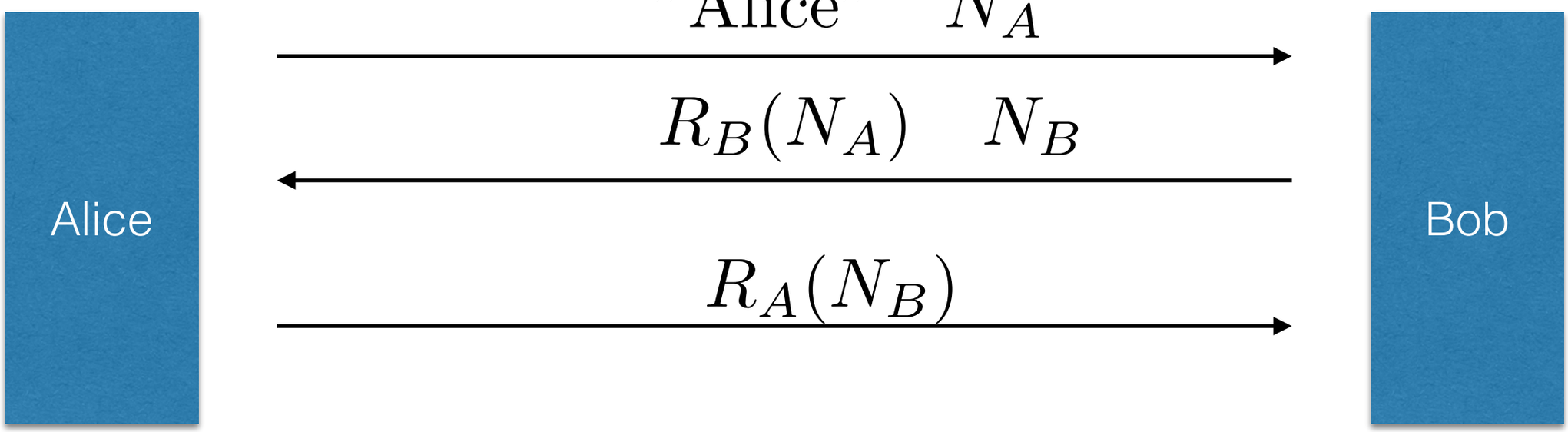
Mutual Authentication



- Mallory then returns to the first session
- Reflects Bob's own answer to her challenge in the second session

Authentication

- Assume that Alice has public key R_A and private key U_A



Alice checks:

$U_B(R_B(N_A)) == N_A$

Bob checks:

$U_A(R_A(N_B)) == N_B$

Key Distribution Centers

- To provide authentication to many different servers:
  - Users & machines have an account with a key distribution center
  - Whenever they need access, they contact the Key Distribution Center (KDC) to obtain credentials for the service
  - Key distribution center implement Single Sign On (SSO)

Key Distribution Centers

- KDC
- For single organization:
  - Use Kerberos, based on Needham Schroeder Protocol
  - Use web-based authentication protocols like Open Authorization OAuth
    - E.g. Twitter OpenID, Google, Amazon, …

Key Distribution Centers

- Needham Schroeder
    1. Alice sends request to KDC, stating her ID and the service she wants to contact

$$N_1, \text{Alice}, \text{Bob}$$

- Nonce in order to label sessions

Key Distribution Centers

- Needham Schroeder

    2. KDC sends Alice a message encrypted with the key shared by her and the KDC
        - The nonce as a session identifier
        - The service name
        - A ticket for the service

- The ticket can only be read by Bob

$$K_A(N_1, \text{Bob}, K_{AB}, K_B(K_{AB}, \text{Alice}))$$

Key Distribution Centers

- Since only Bob can read the ticket

$$K_B(K_{AB}, \text{Alice})$$

- No need to encrypt it with Alice’s key

Key Distribution Centers

- Needham Schroeder
  - The KDC's job is now done
  - Alice and Bob mutually authenticate
  - Alice to Bob:

$$K_B(K_{AB}, \text{Alice}), K_{AB}(N_2)$$

  - with an additional nonce

Key Distribution Centers

- Needham Schroeder
  - Bob unpacks the ticket and then obtains the nonce
  - Proves that he could unpack the ticket and is therefore Bob by performing an arithmetic operation on Alice's nonce.
- Adds a nonce of his own.
- Sends to Alice

$$K_{AB}(N_2 - 1, N_3)$$

Key Distribution Centers

- Needham Schroeder
  - Alice responds by deciphering (proving that she can read the information send by the KDC to her),
  - performing an arithmetic operation on the result
  - and sending it back to Bob

$$K_{AB}(N_3 - 1)$$

Key Distribution Centers

- Alice to KDC: $N_1, \text{Alice}, \text{Bob}$
- KDC to Alice: $K_A(N_1, \text{Bob}, K_{AB}, K_B(K_{AB}, \text{Alice}))$
- Alice to Bob: $K_B(K_{AB}, \text{Alice}), K_{AB}(N_2)$
- Bob to Alice: $K_{AB}(N_2 - 1, N_3)$
- Alice to Bob: $K_{AB}(N_3 - 1)$

Key Distribution Centers

- What happens if the message from Bob to Alice is

$$K_{AB}(N_2 - 1), K_{AB}(N_3)$$

Secure Socket Layer

Thomas Schwarz, SJ

Secure Socket Layer

- 1994: Advent of WWW and potential for commercialization demands security
- Two possibilities:
    - Security layer could alter browsers
        - Because browsers needed to provide security
    - Security layer could change OS and network stack
        - But it would take years for OS to penetrate market
- Needed to get incentives right:
    - Only e-commerce and browser manufacturers were enthusiastic

Secure Socket Layer

- SET - Secure Electronic Transaction
  - Cardholders and merchants need to register with a CA first
  - Customer decides on purchase at web-site
  - Sends payment information: Two linked messages, one for merchant, one for merchant’s bank
  - Merchant forwards card information to bank
  - Bank checks with card issuer for payment authorization
  - Issuer sends authorization to bank
  - Merchant completes order and confirms to customer
  - Merchant captures transaction from their bank
  - Issuer sends invoice to customer

Secure Socket Layer

- SET is more secure than SSL
- But has more stakeholders
  - All need to cooperate,
  - but only some are incentivized
- And so SET was late on the market-place

Secure Socket Layer

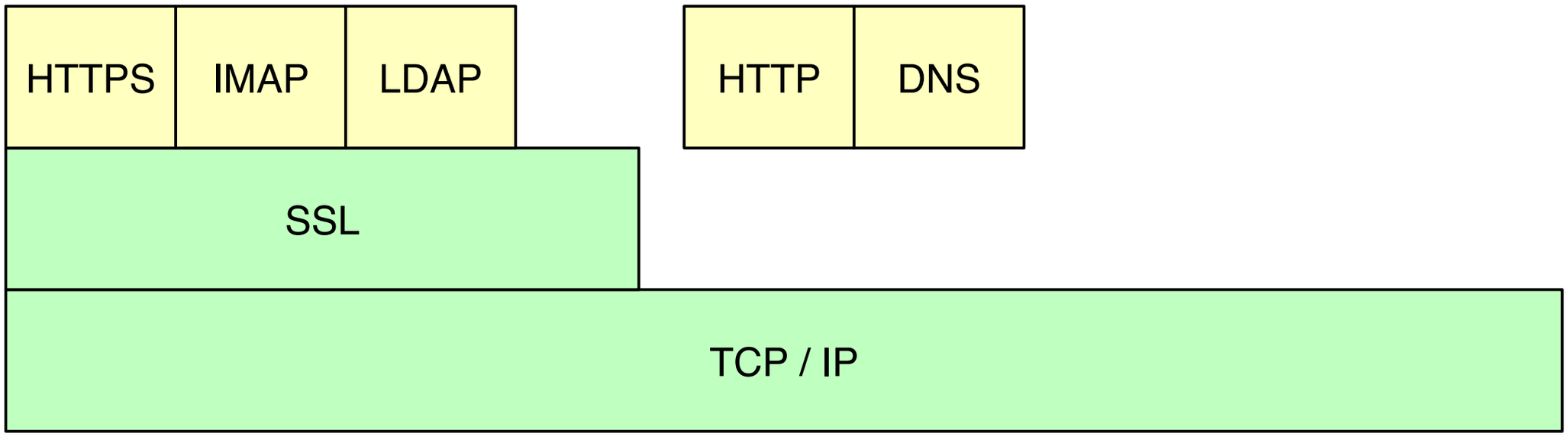
- SSL:
  - Authenticates websites
    - If customer can evaluate certificate
      - So evaluation of certificate has migrated to browser
  - Protects privacy of communication
  - Simulates selling wares through the phone, an established mechanism

Secure Socket Layer

- SSL
  - Provides limited functionality
  - Is still open to fraud
    - But loss rates are manageable
- Needed to be implemented above the OS
  - And changed just browser design

Secure Socket Layer

- Part of the network layer that provides security



SSL-TLS

Alice

Bob

$\text{REQ}, \quad \text{List of supported Crypto}, R_{\text{Alice}}$ →

$\text{Bob's certificate}, \quad \text{Crypto selected}, \quad R_{\text{Bob}}$ ←

Alice invents S, calculates K

$U_{\text{Bob}}(S), \quad \text{hash}_K(\text{previous two messages})$ →

$U_{\text{Bob}}(S), \quad \text{hash'}_K(\text{previous two messages})$ ←

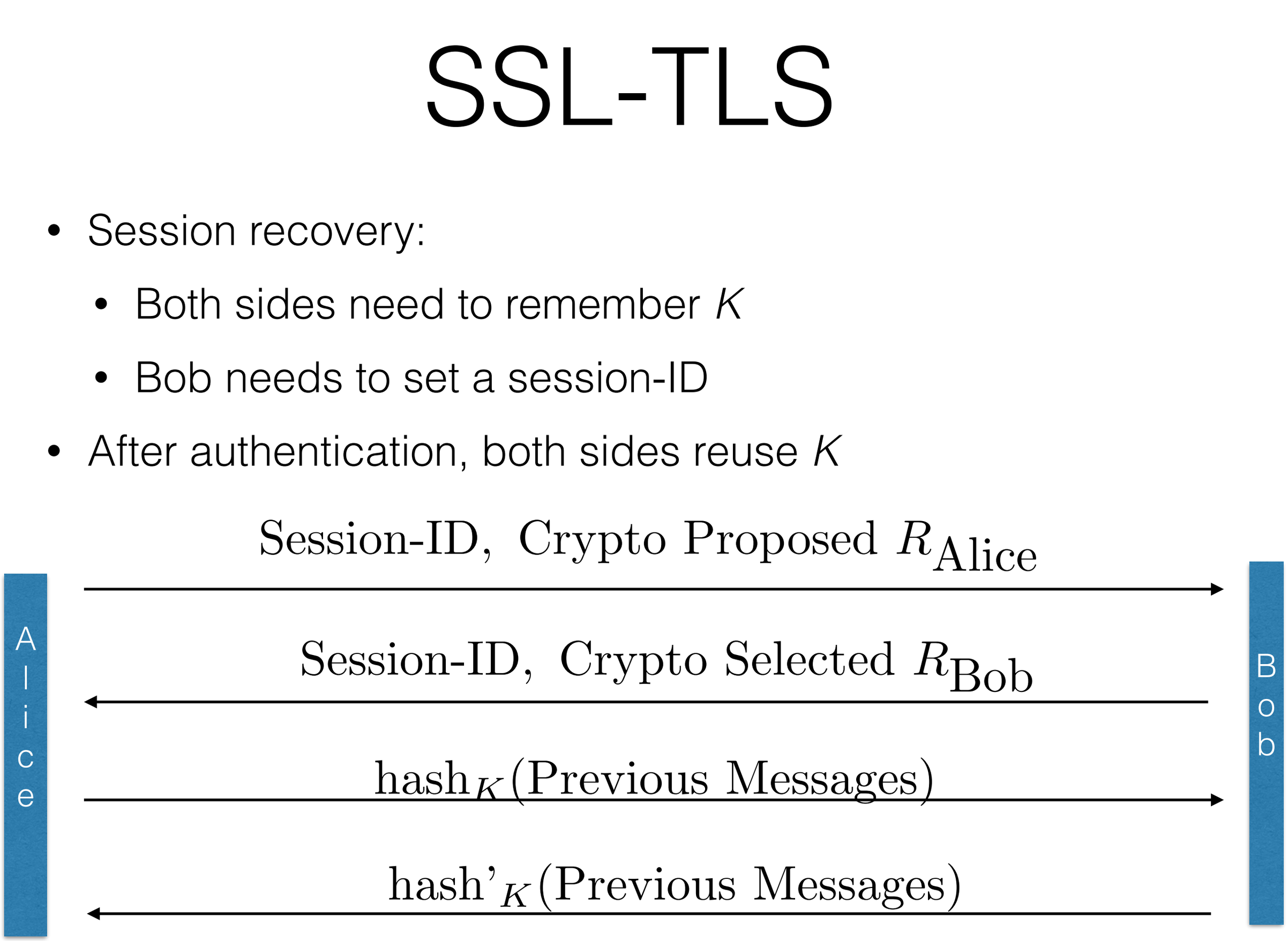
$$K = f(S, R_{\text{Alice}}, R_{\text{Bob}})$$

SSL-TLS

- Session key *K* derived from random values by both parties
  - S — pre-master secret
  - K — master secret
- Only Bob is authenticated
  - Which means client has to authenticate within the application in order to process payment information

SSL-TLS

- Session recovery:
  - Both sides need to remember *K*
  - Bob needs to set a session-ID
  - After authentication, both sides reuse *K*



SSL-TLS

- Crypto-protocol negotiation
  - All protocols are defined and have an identity number
- Decision:
  - Version 2: Bob sends a list of protocols in message 2 that he deems acceptable and Alice sends her selection in message 3
  - Version 3: Bob selects a protocol in message 2

SSL-TLS

- Downgrade attack in Version 2
  - There is no integrity protection in the initial exchange and a MitM could change Bob's message to select a weak crypto
- Truncation attack:
  - MitM inserts a TCP code indicating that the session has finished
  - Recipient does not pick up the rest of the message
  - Version 3 uses a closing handshake to terminate sessions properly

SSL-TLS

- Export Rules:
  - SSL version 2 has keys of 128b
  - Export rules prohibited keys of this length
  - Export versions:
    - In her second message, Alice sends an *S* of 40b protected by public key and the other 88b in plain text

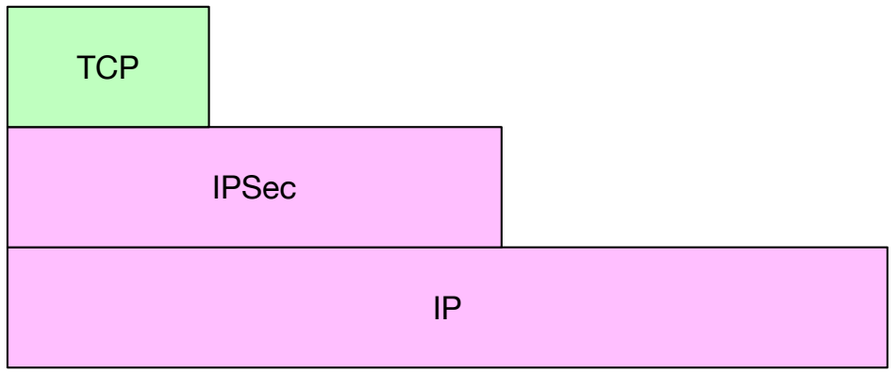
IPSec

- Goal is to make IP secure
- Started in 1995 at IETF
- Is now supported by all major OS
- Relies on IP version 6
- Which is still not universal
  - Applications should be oblivious to use of IPSEC
- Change OS, leave App design alone

Security in the Networking Stack

- Operating at a higher levels allows attacks at lower levels
  - Example: SSL is above TCP-layer
    - Adversary can introduce TCP packages with given packet numbers
    - OS drops genuine TCP packages with those packet numbers
    - SSL is missing packets and has to close

IPSec



- IPSec
  - Tries to make IP secure
    - Sits above IPv6 and below TCP-layer
  - Uses two distinct phases:
    - Session key establishment phase
      - Manual configuration
      - IKE (Internet key exchange)
    - Use phase

IPSec

- Uses Security Associations (SA)
    - One-way connection
- So, Alice and Bob have each their own SA
- Cryptographically protected connection
- All IPSec packages have a Security Parameter Index (SPI)
- Everybody has a SA database
    - SPI is used as an index into the SA database
- If Alice sends a packet to Bob, the SPI tells Bob what SA to use and the SA tells Bob what to do with the package
- For multicast: SPI set by sender instead of recipient

IPSec

- SA database
  - Alice wants to send Bob (identified by IP address) a package
  - Alice looks for Bob in her SA database
  - Database returns:
    - SPI
    - Key
    - Algorithms to use
    - Sequence numbers
    - …

IPSec

- Bob receives a packet
  - Bob retrieves the SPI from the package
  - Looks for the corresponding entry
  - Bob gets
    - key
    - algorithm
    - sequence number

IPSec

- Security policies database
  - IPSec has a database with contents similar to that of a firewall
    - Based on IP header source, destination, TCP port … determines whether
    - to drop the packet
    - to resend the packet
    - to accept with IPSec protection
    - to protect with IPSec and how

IPSec

- Use phase works in two modes:
  - Authentication Header AH
  - Encapsulating Security Payload: ESP
- Only integrity protection
  - Uses AH header
  - Proves that messages have been received completely from a given device
- Optional integrity protection and Optional Privacy
  - Uses ESP header
  - Allows packets to be encrypted
  - Allows integrity of packets to be checked

IPSec

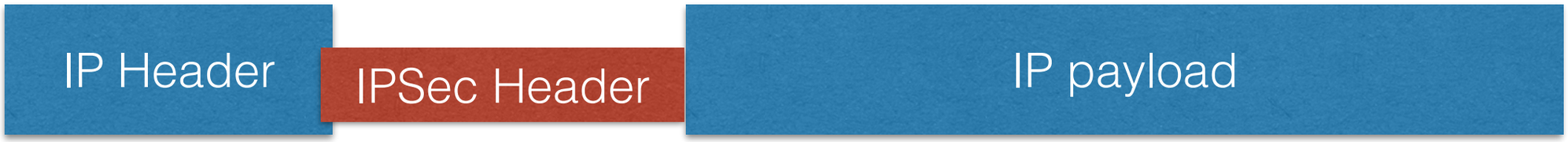
- Difference between AH integrity protection and ESP integrity protection
  - AH allows inspection by firewalls and routers

IPSec

- IPSec has two modes:
  - Transport
    - Just adds an IPSec header to IP-packet
  - Tunnel
    - Creates a new IP header and a new IPSec header

IPSec

Transport



Tunnel

IPSec

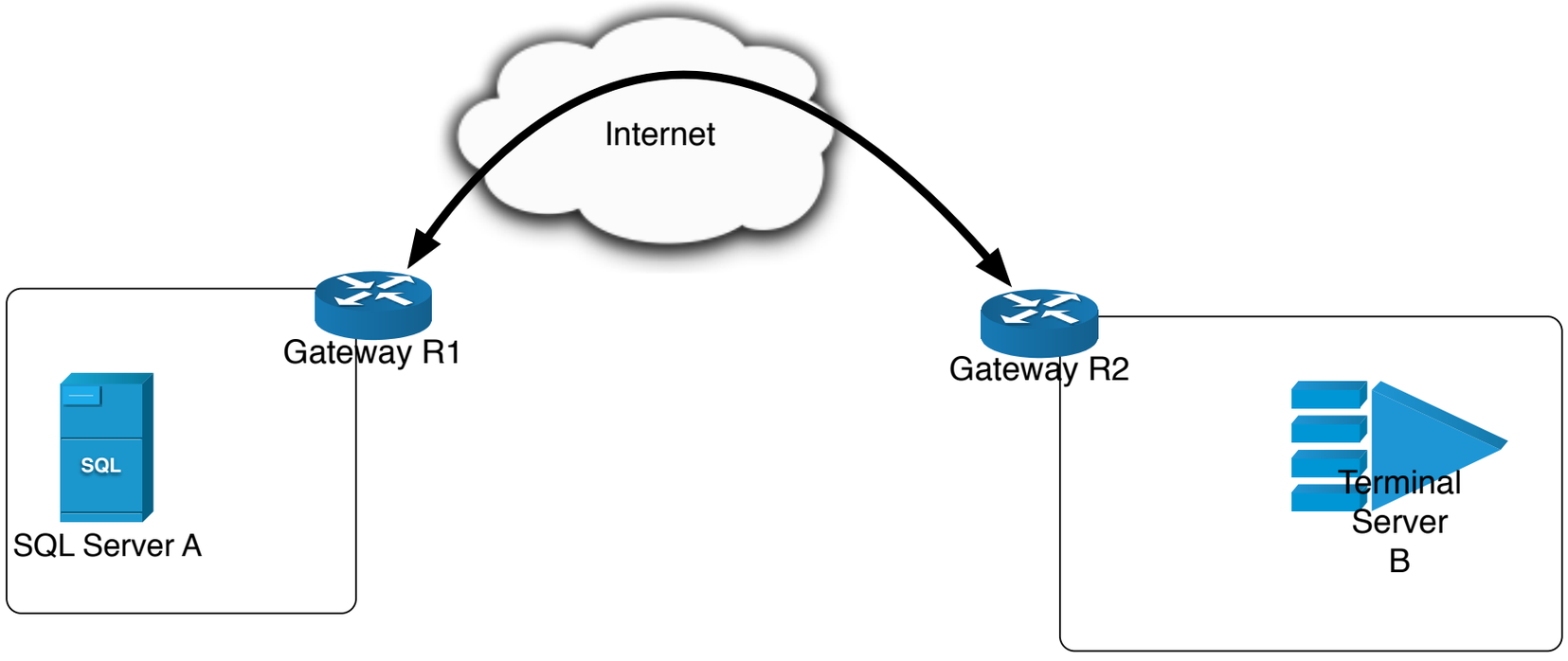
Original Packet	IPsec Package in Transport Mode	IPSec Package in Tunnel Mode
IP header | rest	IP header | IPsec header | rest	new IP hdr | IPSec | IP header | rest

IPSec

- Most frequent mode for IPSec:
  - Virtual Private Networks

IPSec in Tunnel Mode

IP: src = R1, dst = R2 | ESP | {IP: src=A, dst=B | payload}



IPSec

- Authentication Header

1B	1B	2B	4B	4B	variable
Next header	Payload length	Unused	SPI	Sequence Number	Authentication data

- Next header: position of protocol field of encapsulated package
- Payload length: Size of AH header in words.
- SPI (Security Parameter Index)
- Sequence number: Used by AH to recognize replayed packages. Not identical with TCP package number.
- Authentication data: Cryptographic integrity check on the payload data.

IPSec

- Authentication Header
  - Uses a MAC to protect IP header fields against alteration
  - Some *mutable* attributes are excluded because they are changed by routers
    - Fragmentation, Time-to-live
  - Some attributes change predictably
    - Source routing: IP destination changes
  - Predictable attributes are protected by AH

IPSec

- ESP Header

4	4	var.	var.	var.	1	1	var.
SPI	Sequence number	IV	data	padding	padding length	Next header / protocol type	Authentication data

  - SPI
  - Sequence Number (same as for AH)
  - IV Initialization Vector (used by some cryptographic algorithms
  - Data: protected data, possibly encrypted (e.g. TCO header)
  - Padding: needed to make data multiple of block size.
  - Padding length
  - Next header: Protocol field in IPv4 or next header in IPv6
  - Authentication data: Cryptographic integrity check.

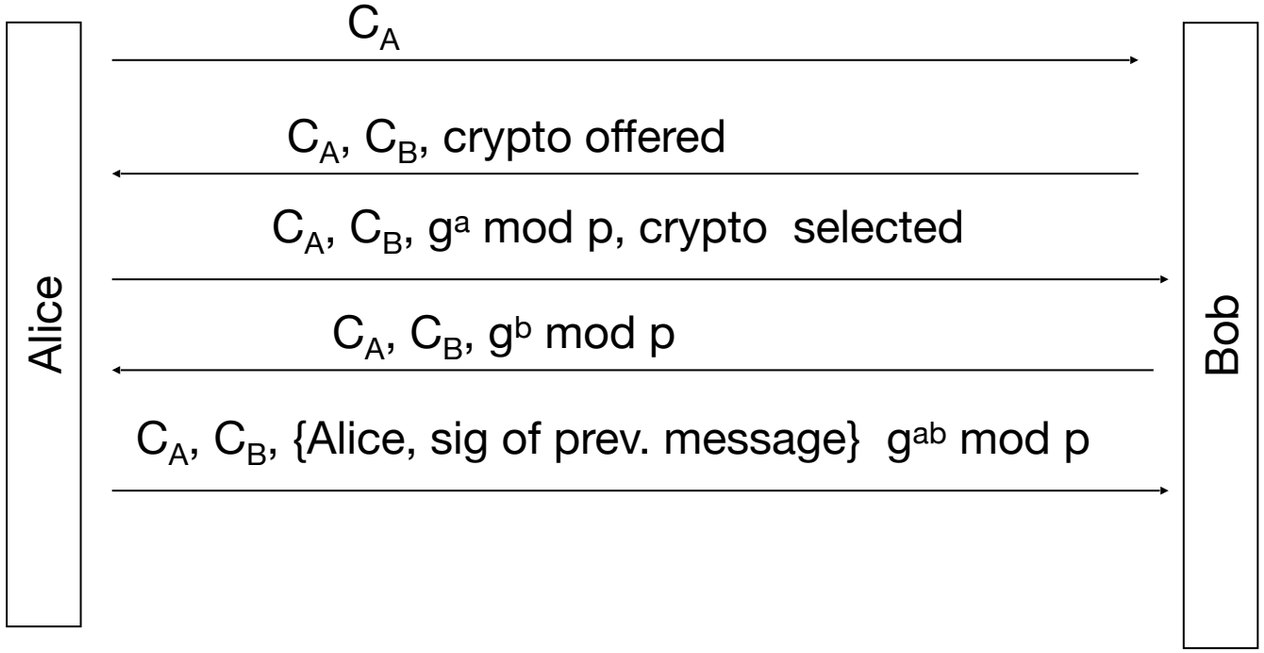
IPSec

- Why AH
  - AH is not subject to export control
    - No longer important
  - AH protects IP header
    - Which is not very useful

IPSec

- Internet Key Exchange (IKE)
  - Serves to establish a session key
  - Serves to establish a SA
  - Derived from Photuris and SKIP protocols

IPSec



Photuris

SKIP

- SKIP — Simple Key Management for Internet Protocols
  - Predecessor of IKE and very common in the 90s
  - Public Diffie-Hellman keys

SKIP

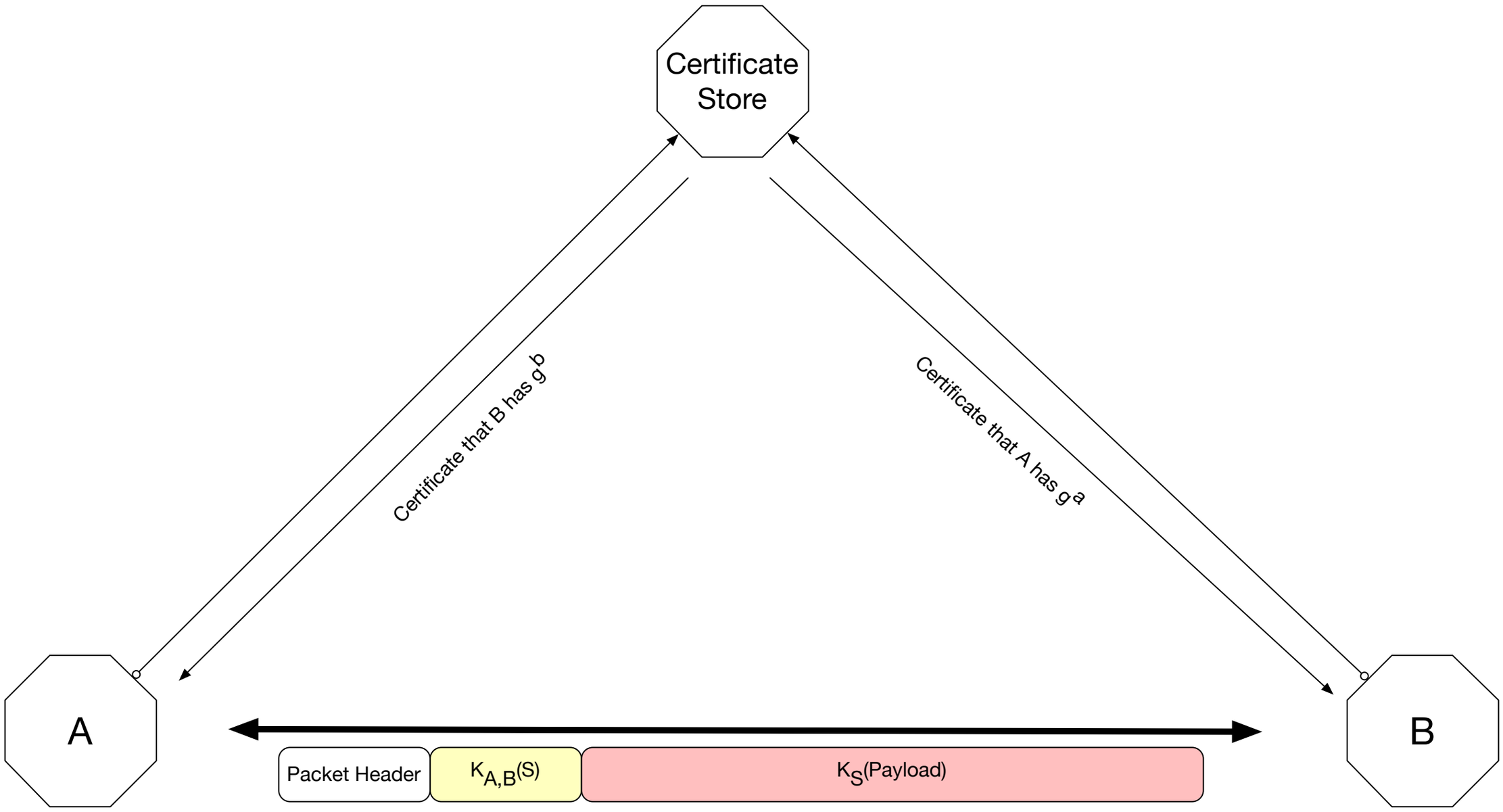
- Public Diffie-Hellman keys
  - Everybody agrees on the algebraic structure, such as $\mathbb{Z}_p$
  - Everybody agrees on the generator $g \in \mathbb{Z}_p$
  - Everyone has a secret number *a* and publishes g^a
    - Manual distribution
    - Trusted directory
    - Signed by owner
    - Part of certificate
- Two parties use the Diffie-Hellman protocol to agree on a key

$$K_{\mathrm{A,B}} = g^{ab}$$

SKIP

- Packets contain
  - Packet key
  - Packet key encrypted with the common Diffie-Hellman key (KEK)

SKIP


Certificate Store
Certificate that B has g^b
Certificate that A has g^a
A
B
Packet Header
$K_{A,B}(S)$
K_S(Payload)

IPSec

- History:
  - SKIP and Photuris are two different proposals in 1995
  - Internet Security Association and Key Management Protocol ISAKMP 1998
    - NSA framework for UDP
- Oakley protocol is part of ISAKMP
- SKEME (Security Key Exchange Mechanism) 1996
- IKE (RFC 2407) 1998
  - Uses ideas of Oakley and SKEME
  - Uses ISAKMP as framework
  - 150 pages of documentation

IPSec

- Key establishment
  - Can be manual
  - Can use IKE

IKE

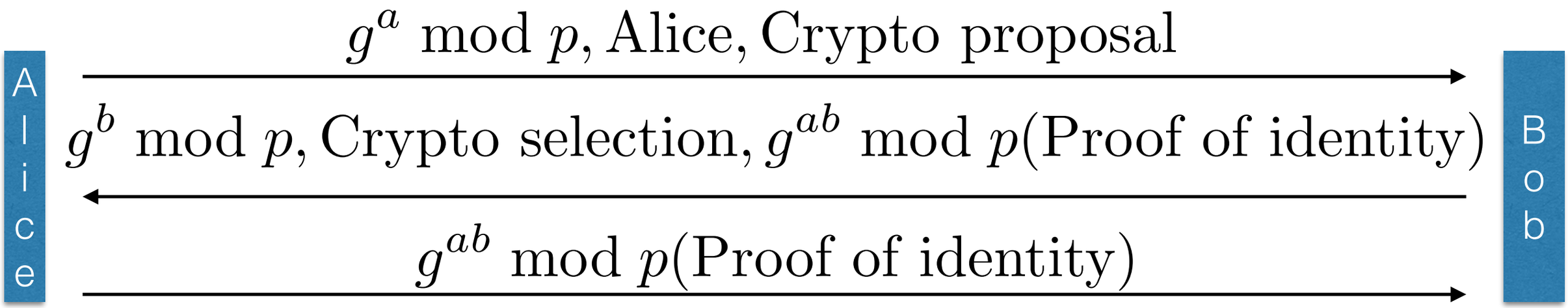
- Phase 1:
  - Mutual authentication and session key establishment
    - Uses identities and secrets (such as public keys or shared secrets)
- Phase 2:
  - SA (Security Association) establishment

IKE

- Idea of phases:
  - Can use same authentication to establish SAs
    - You need at least two per conversation
  - Authentication might be use by more than a single protocol
  - Session key and global key changes are more simple
  - Can establish connections with different security properties
    - E.g. not guarantee confidentiality for bulk data

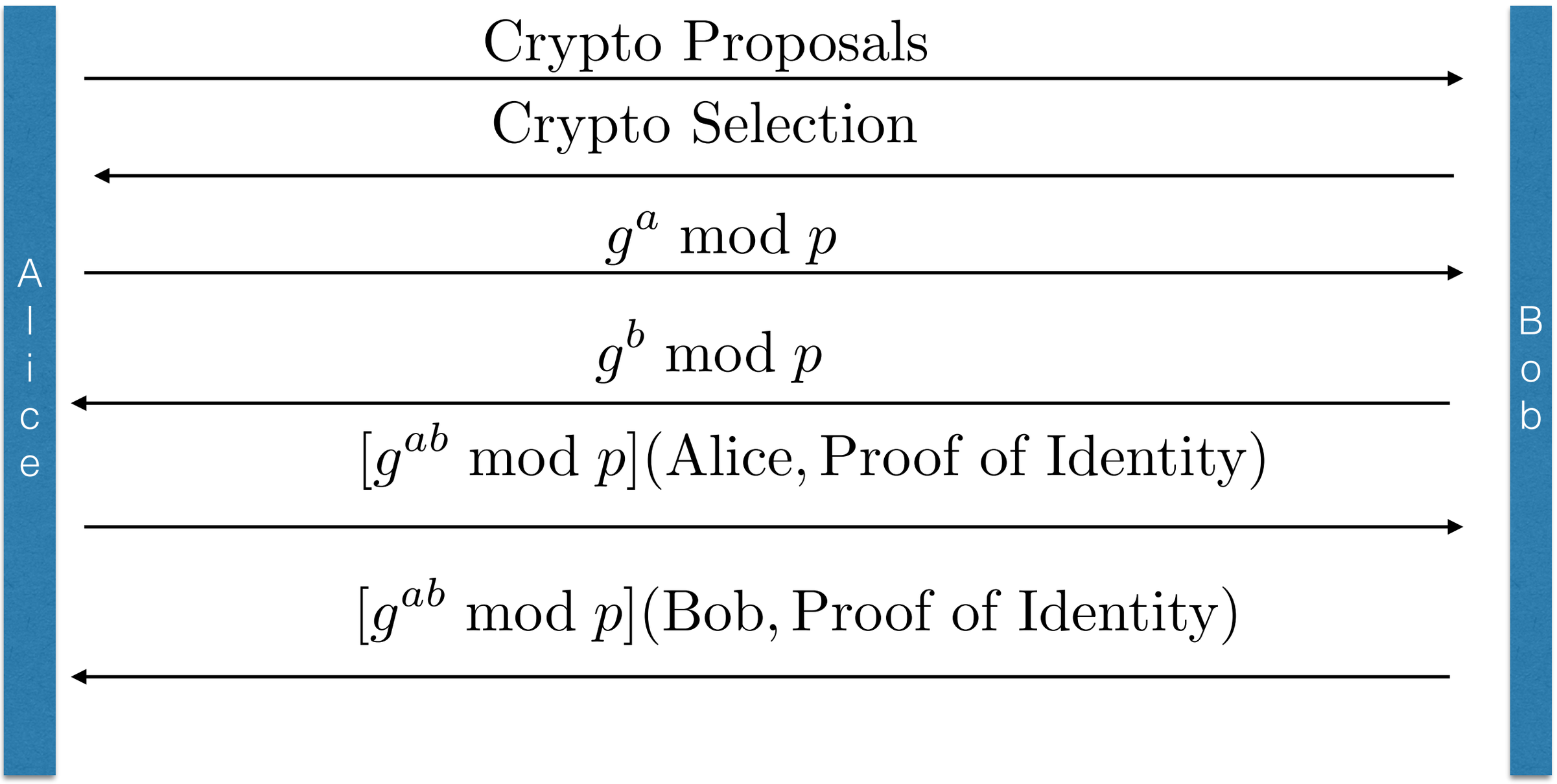
IKE

- IKE-Phase 1:
  - Several modes
  - Aggressive mode



IKE

- IKE-Phase 1: Principal mode



IKE

- IKE phase 1 uses one of these possibilities:
  - Pre-shared key
  - Public key for encryption
    - Two versions of protocol
      - Proof of identity based on key-type
      - Cookies
        - Uses two cookies interchanged in the first two messages (as in Photuris)
          - Which are useless because Bob needs to maintain state
        - IKE interchange is identified by two additional nonces

IKE

- Phase Two: Setting up SAs
  - Quick mode
    - Connected to a phase 1 interchange
    - Uses the Photuris cookies generated in phase 1

IKE

Alice

Bob

$X, Y,$ Crypto-protocols proposed, $\mathrm{SPI}_A, \mathrm{nonce}_A$ →

← $X, Y,$ Crypto-protocols accepted, $\mathrm{SPI}_B, \mathrm{nonce}_B$

$X, Y,$ ack →

SSH/SCP/SFTP/FISH

- Protocol suite implemented on top of OS
  - Provide
    - Authentication, Integrity, Confidentiality
  - SCP/SFTP/FISH:
    - Moving files in a secure manner
  - SSH:
    - Secure remote command execution
  - SSHFS:
    - Mounting a remote directory in a secure manner

SSH

- SSHv1
  - Client contacts sender at port 22
  - Client and Server interchange their ssh-versions
  - Change to a packet based protocol
    - Packets are:
      - 4B length
      - 1-8B padding
      - 1B packet code
      - payload
      - 4B for identity verification

SSH

- SSHv1
  - Server answers with
    - Guest key
    - Server key
    - Cookie (8 random bytes)
    - List of protocols (encryption, compression, authentication)
- Both calculate a session identifier of 128b

SSH

- SSHv1
  - When the client receives the guest key, the client looks for the key in the known-host database
    - If the guest key is there, continues
    - If the guest key is not there: asks the user
    - If the guest key is different: alarms the user to a possible fraud
- Assumption: You trust the first time around

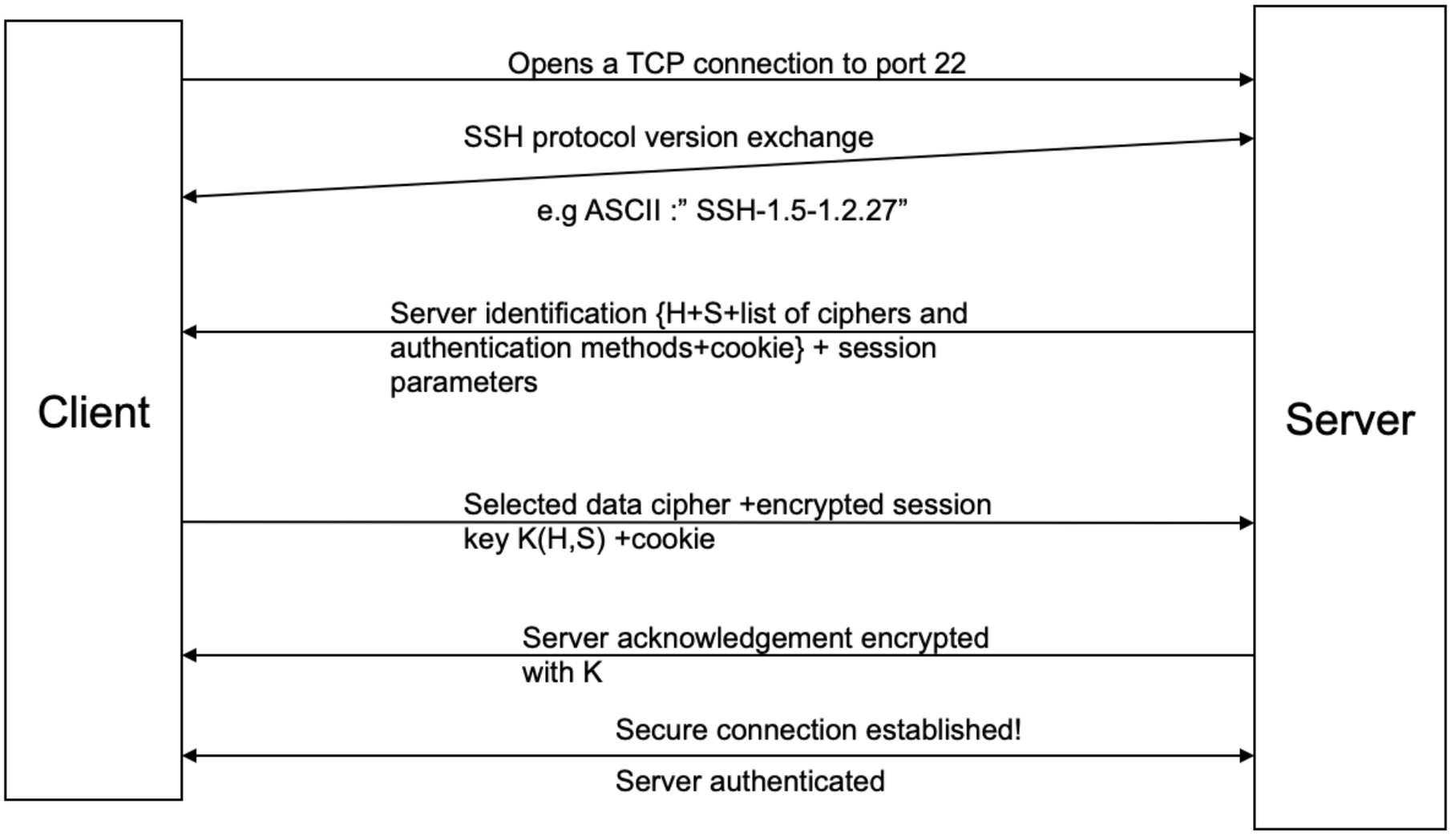
SSH

- Client now generates a random session key
- Client sends crypto-protocol list and this session key encrypted by
  - Server key
  - Public key of host
- Both sides now use the session key
- Server is now authenticated
  - SSH tries Kerberos, Rhosts, RhostsRSA, Public Key , TIS, password
- At this point, the client relies on the identity of the sender
- Server authenticates client at this point, usually through password

SSH

- Changes to version 2:
  - Removed some vulnerabilities
  - is made modular
  - supports more protocols
  - has better integrity protection
  - Restricts user authentication to
    - Public key: DSA, RSA, OpenPGP
    - IP-address
    - Password

SSHv1

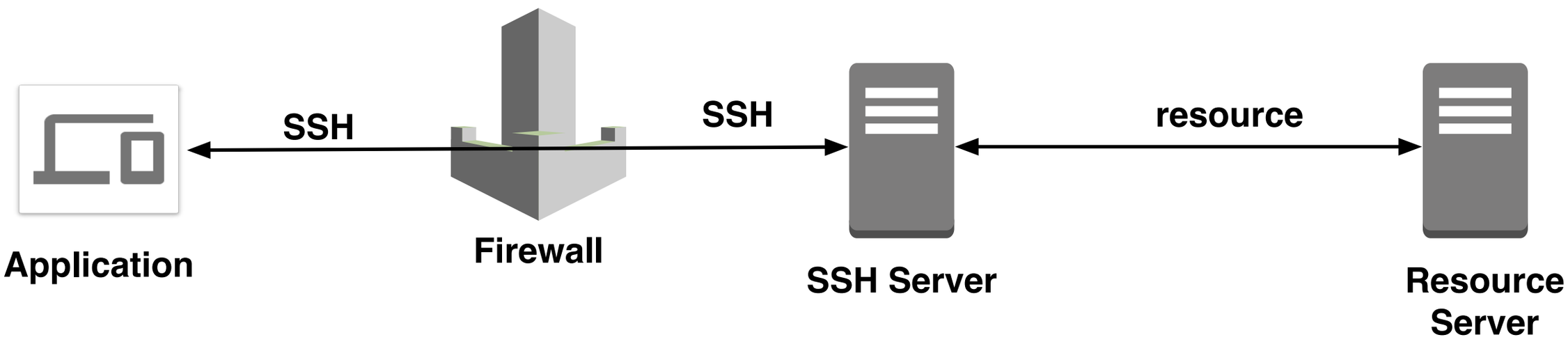


H – host key ; S- Server Key ; cookie- sequence of 8 random bytes; K- session key

SSH Tunneling

- SSH Port Forwarding:
  - Local SSH client establishes a connection with a remote SSH server
  - Connection is forwarded to a resource within the trusted internal network

```
ssh -L local_port:destination_server_ip:remote_port ssh_server_hostname
```

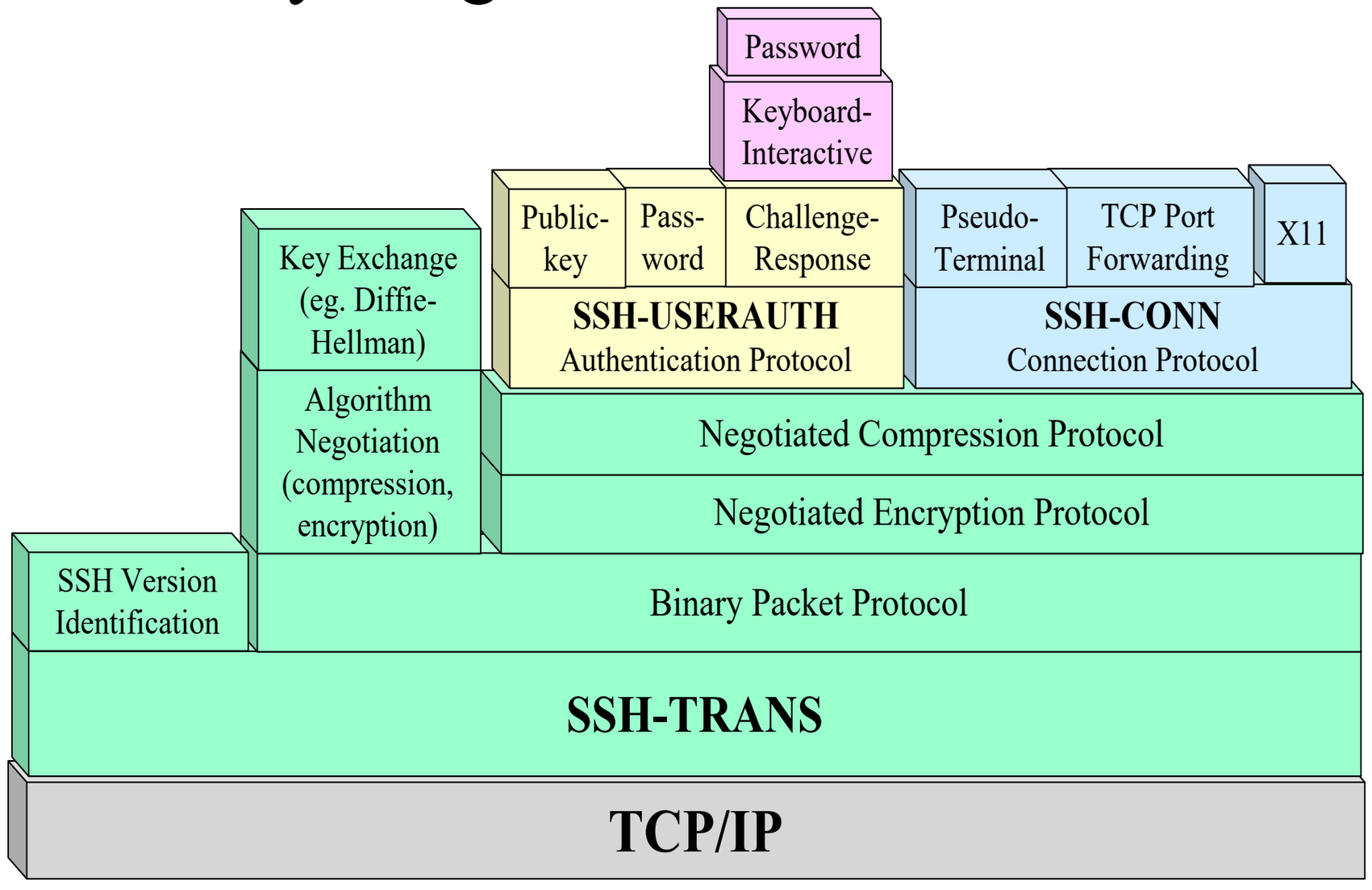


SSH Tunneling

- Example:
  - SOCKS — Socket Secure
    - Facilitates communication through a fire-wall
    - Socks proxy server creates a TCP connection on behalf of a client
      - Exchanges packets between client and actual server
    - Not accessible to protocols below TCP, such as ARP, ping, Nmap, …
    - But HTTP, HTTPS, POP3, SMTP, FTP, …

SSH-2

Layering of SSH-2 Protocols



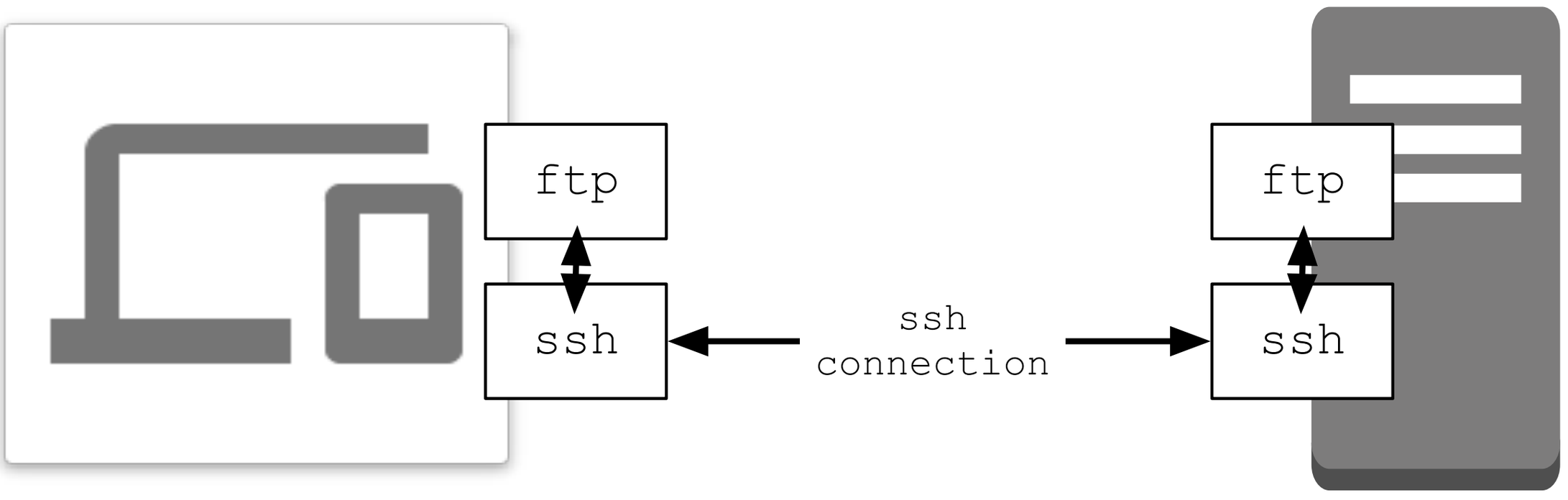
SSH-2

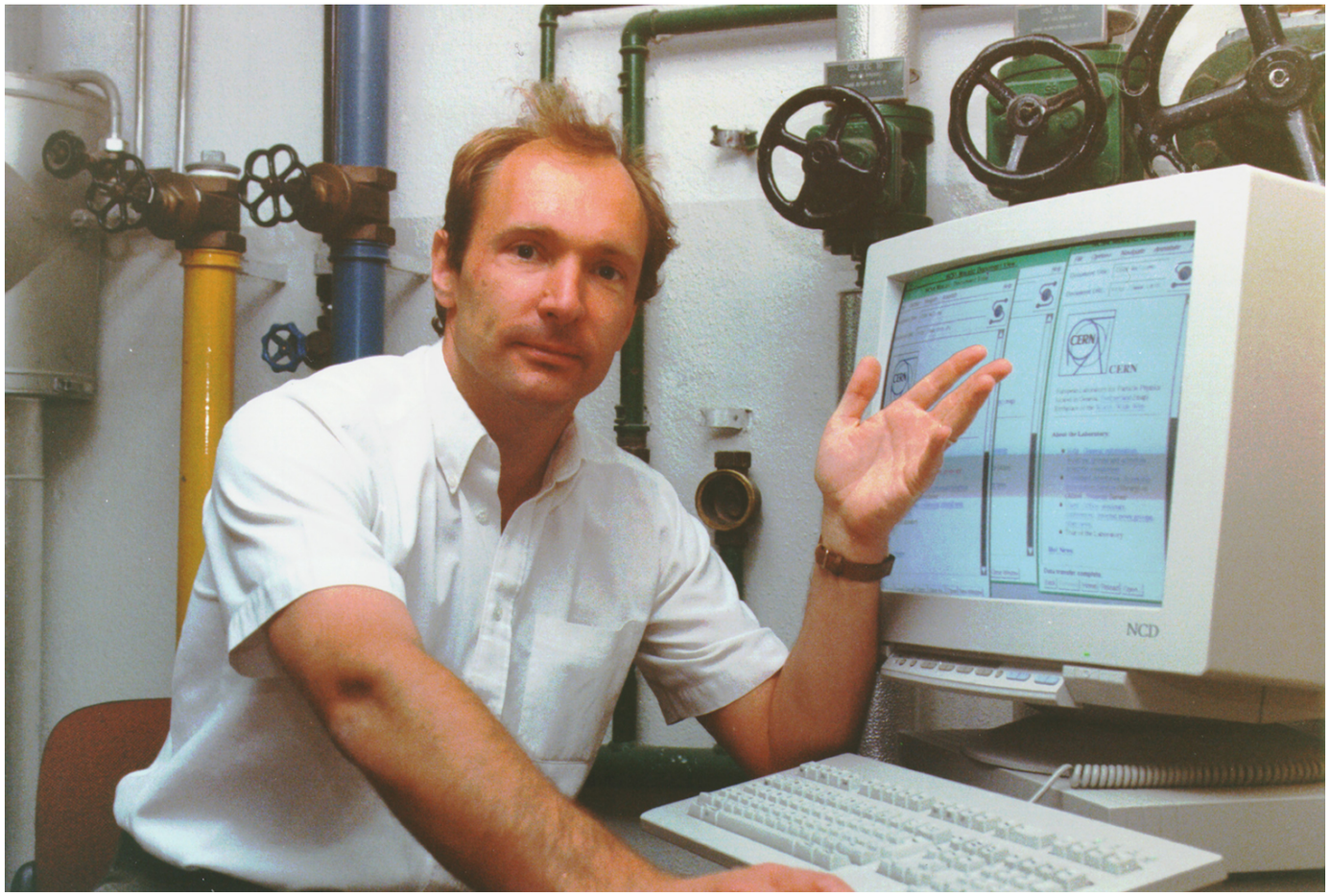
- SSH transport layer
  - initial connection, packet protocol, server authentication, basic encryption and integrity services
- SSH authentication layer
  - Uses public key, host based, or password
- SSH connection protocol
  - Permits different services uses an SSH channel

SSH Tunneling

- Use dynamic port forwarding on SSH client to create a SOCKS proxy
  - `ssh –D local_port ssh_server_hostname`

SSH Tunnelling

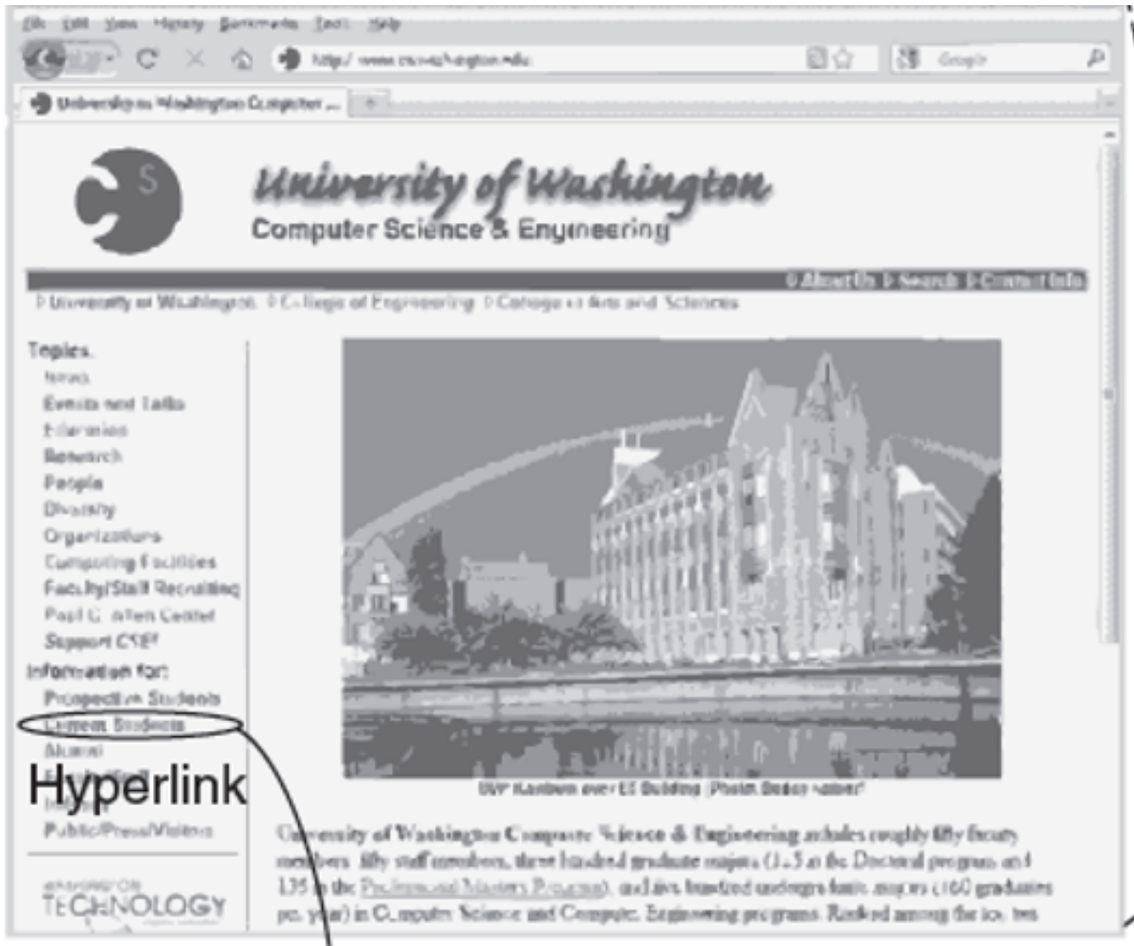




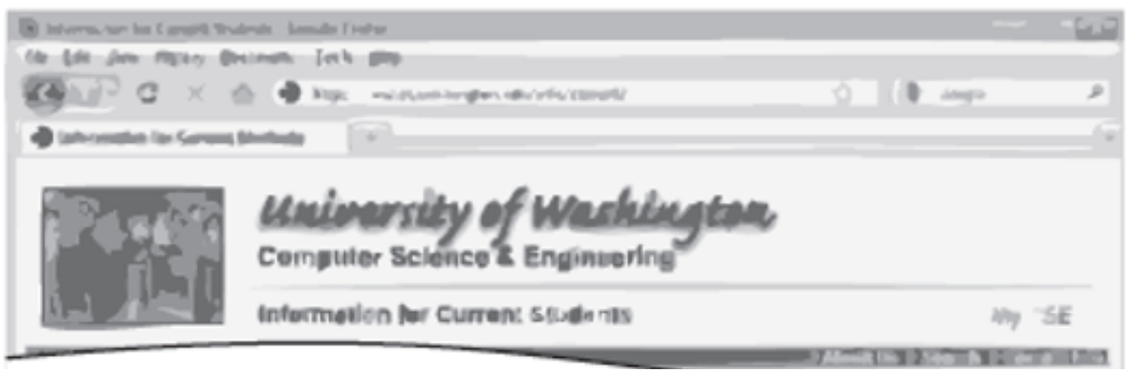
Tim Berners-Lee

WWW

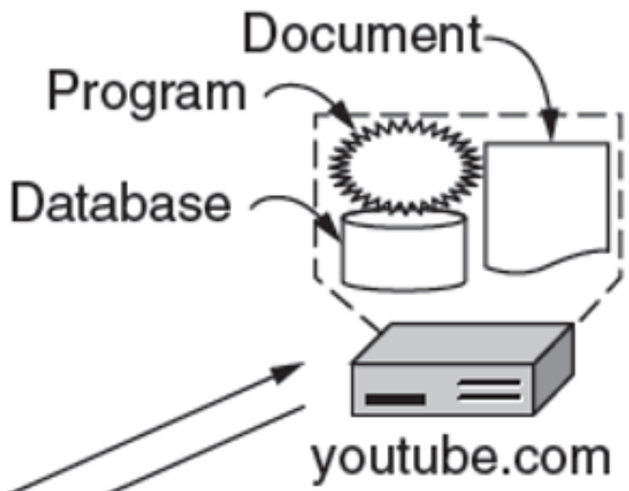
Thomas Schwarz, SJ



Web page



Web browser



HTTP Request

HTTP Response

Web server
www.cs.washington.edu

google-analytics.com

WWW

Pages are named with URLs (Uniform Resource Locators)

- Example: http://www.phdcomics.com/comics.php

Protocol Server Page on server

Name	Used for	Example
http	Hypertext (HTML)	http://www.ee.uwa.edu/~rob/
https	Hypertext with security	https://www.bank.com/accounts/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Local file	file:///usr/suzanne/prog.c
mailto	Sending email	mailto:JohnUser@acm.org
rtsp	Streaming media	rtsp://youtube.com/montypython.mpg
sip	Multimedia calls	sip:eve@adversary.com
about	Browser information	about:plugins

WWW

- 1989: Tim Berners-Lee @ CERN
  - Access documents freely through hyperlinks
- Hypertext:
  - Memex: Vannevar Bush 1940s: associative indexing
  - Douglas Engelbart 1960s: oN-Line System (NLS) for office automation
    - creates link between different documents and services
    - Uses mouse and window manager

WWW

- Ted Nelson 1963 — : Xanadu — a “docuverse” where all information ever created will be available to everyone else

WWW

- Hypertext:
  - Links connect a universe of documents
  - Later became hypermedia
- Markup language
  - Originally for annotating an electronic document
    - Typesetting in troff
  - Standard Generalized Markup Language SGML ISO 8879:1986
  - HTML (1991)
  - XML (1998)

WWW

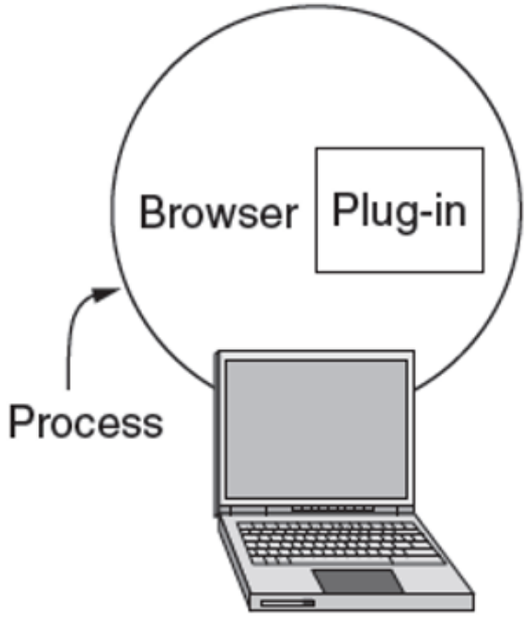
- Uniform Resource Locator URL
  - Protocol field + Canonical Resource Name
    - `<scheme>://<host>:<port>/<path>;<parameters>?<query>#<fragment>`
- Uniform Resource Name URN
  - Globally unique, persistent identifier
    - Independent of location
- Uniform Resource Identifier
  - Collection of URLs and URNs

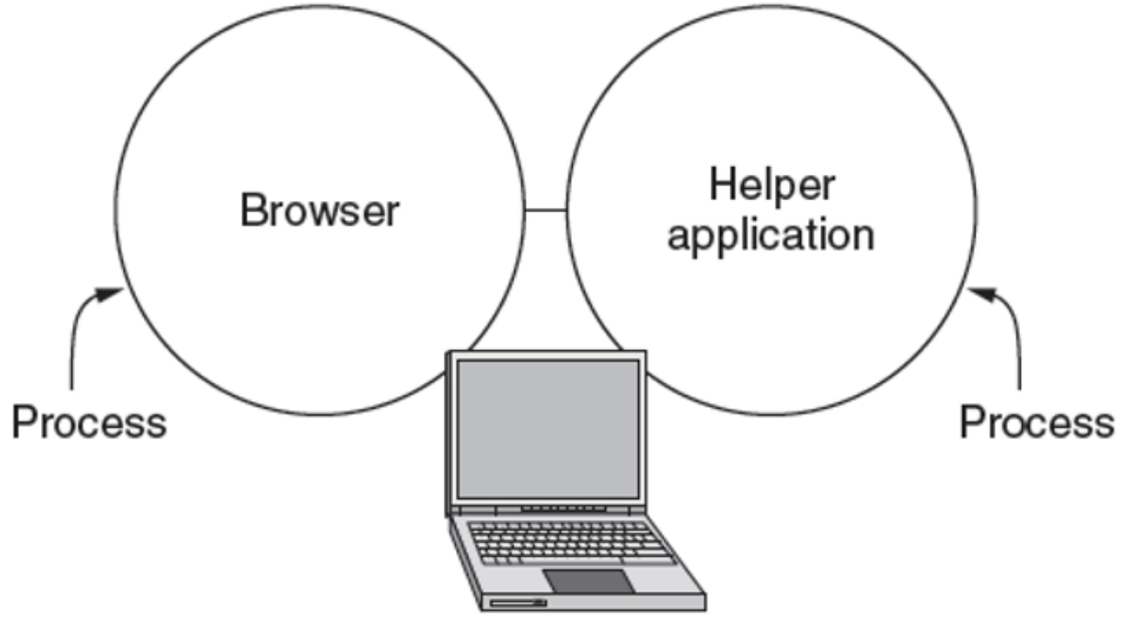
WWW

- Steps a client (browser) takes to follow a hyperlink:
  - Determine the protocol (HTTP)
  - Ask DNS for the IP address of server
  - Make a TCP connection to server
  - Send request for the page; server sends it back
  - Fetch other URLs as needed to display the page
  - Close idle TCP connections
- Steps a server takes to serve pages:
  - Accept a TCP connection from client
  - Get page request and map it to a resource (e.g., file name)
  - Get the resource (e.g., file from disk)
  - Send contents of the resource to the client.
  - Release idle TCP connections

WWW

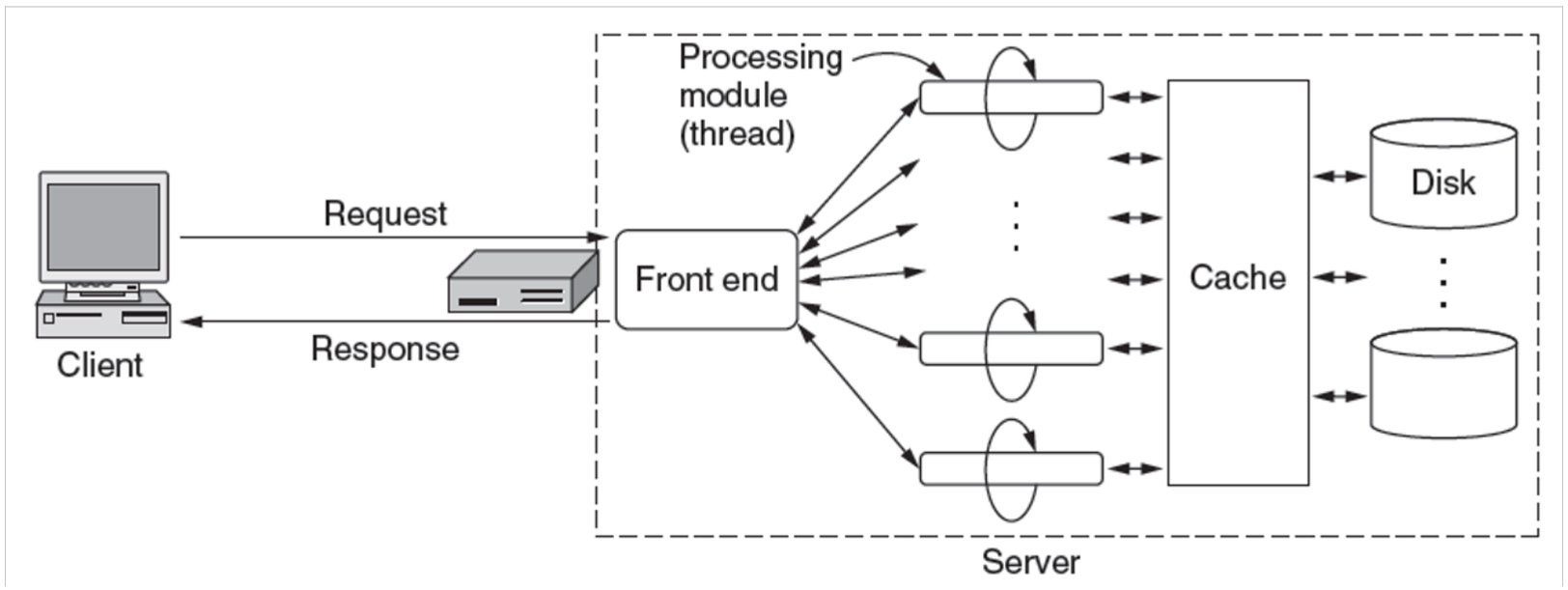
- Content type is identified by MIME types
  - Browser takes the appropriate action to display
  - Plug-ins / helper apps extend browser for new types





www

- Performance at the web-server side
  - Caching
  - Multiple threads
- Frontend



WWW

- WWW was stateless, but e-commerce needs state
- Cookies support stateful client/server interactions
  - Server sends cookies (state) with page response
  - Client stores cookies across page fetches
  - Client sends cookies back to server with requests
- Examples:

Domain	Path	Content	Expires	Secure
toms-casino.com	/	CustomerID=297793521	15-10-10 17:00	Yes
jills-store.com	/	Cart=1-00501;1-07031;2-13721	11-1-11 14:22	No
aportal.com	/	Prefs=Stk:CSCO+ORCL;Spt:Jets	31-12-20 23:59	No
sneaky.com	/	UserID=4627239101	31-12-19 23:59	No

WWW

- Static webpages
    - Simple files with same contents each viewing
- Can be interactive
    - HTML with embedded contents
    - Forms that gather input
    - Style sheets
    - Vector graphics, …

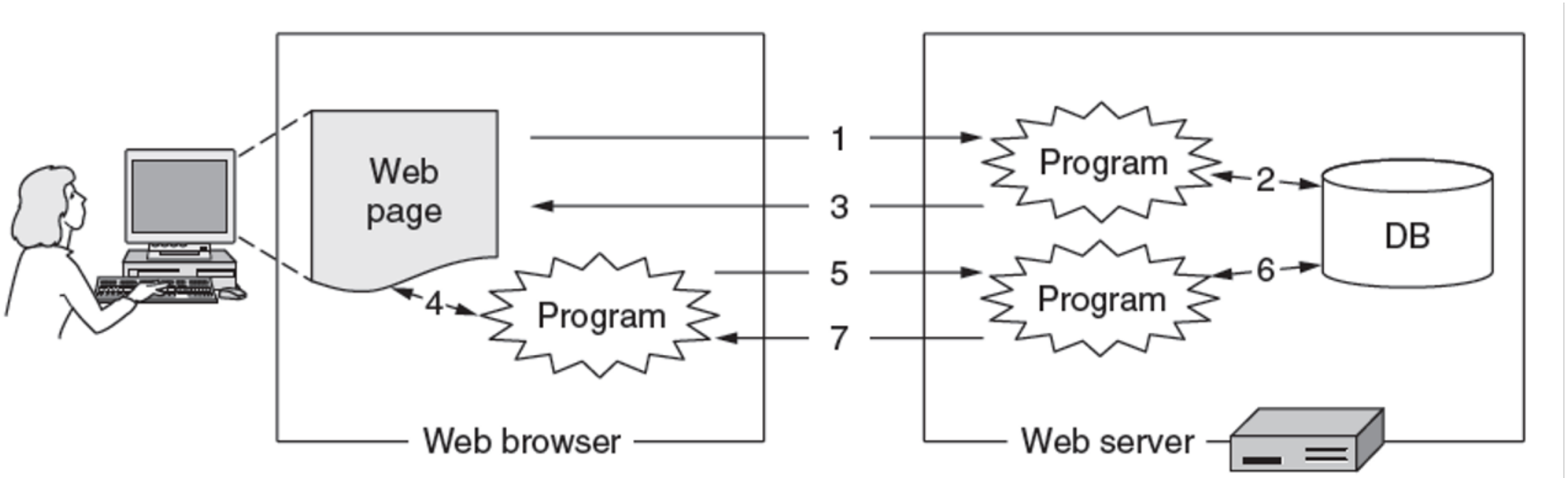
www

- Evolution of html

Item	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0	HTML 5.0
Hyperlinks	x	x	x	x	x
Images	x	x	x	x	x
Lists	x	x	x	x	x
Active maps & images		x	x	x	x
Forms		x	x	x	x
Equations			x	x	x
Toolbars			x	x	x
Tables			x	x	x
Accessibility features				x	x
Object embedding				x	x
Style sheets				x	x
Scripting				x	x
Video and audio					x
Inline vector graphics					x
XML representation					x
Background threads					x
Browser storage					x
Drawing canvas					x

WWW

- Dynamic pages are generated by programs running at the server (with a database) and the client
  - E.g., PHP at server, JavaScript at client
  - Pages vary each time like using an application



WWW

Web page that gets form input and calls a server program

```
<html>
<body>
<form action="action.php" method="post">
<p> Please enter your name: <input type="text" name="name"> </p>
<p> Please enter your age: <input type="text" name="age"> </p>
<input type="submit">
</form>
</body>
</html>
```

PHP server program that creates a custom Web page

PHP calls

```
<html>
<body>
<h1> Reply: </h1>
Hello <?php echo $name; ?>.
Prediction: next year you will be <?php echo $age + 1; ?>
</body>
</html>
```

Resulting Web page (for inputs “Barbara” and “32”)

```
<html>
<body>
<h1> Reply: </h1>
Hello Barbara.
Prediction: next year you will be 33
</body>
</html>
```

www

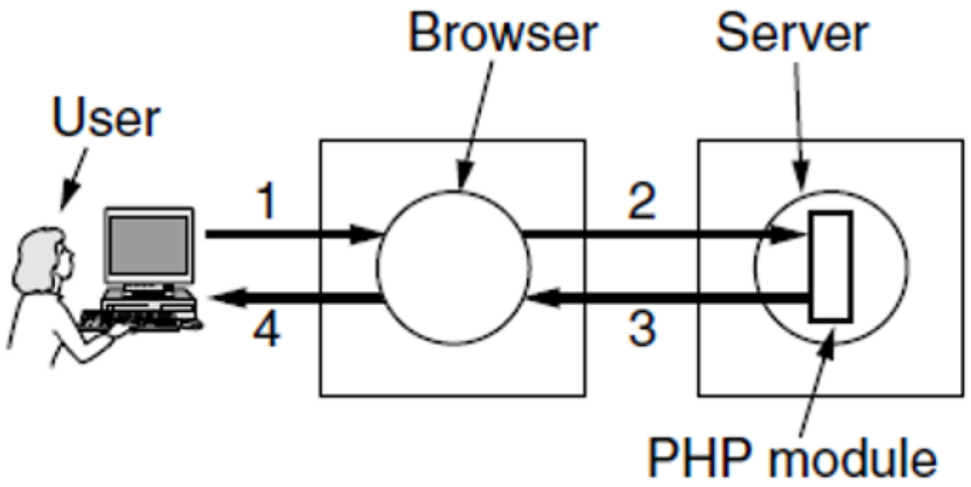
JavaScript program produces result page in the browser

```
<html>
<head>
<script language="javascript" type="text/javascript">
function response(test_form) {
    var person = test_form.name.value;
    var years = eval(test_form.age.value) + 1;
    document.open();
    document.writeln("<html> <body>");
    document.writeln("Hello " + person + ".<br>");
    document.writeln("Prediction: next year you will be " + years + ".");
    document.writeln("</body> </html>");
    document.close();
}
</script>
</head>
```

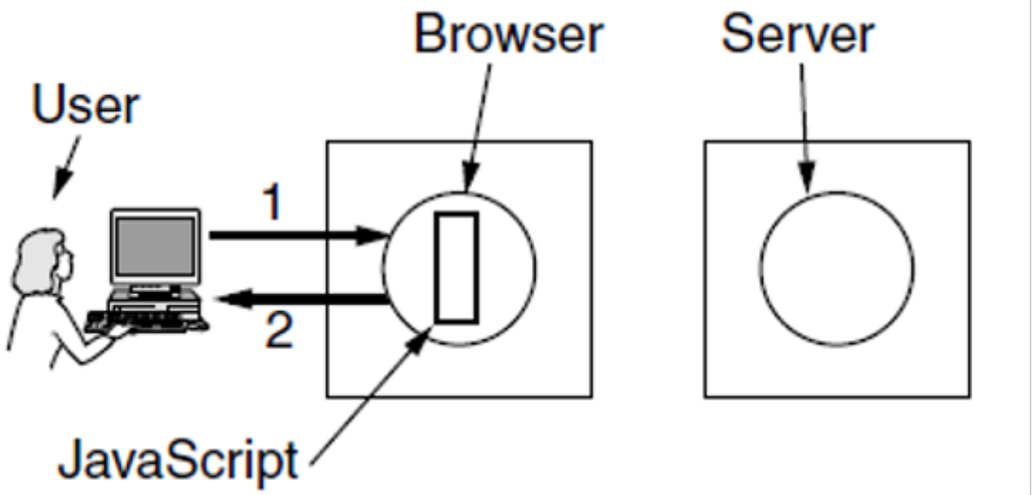
First page with form, gets input and calls program above

```
<body>
<form>
Please enter your name: <input type="text" name="name">
<p>
Please enter your age: <input type="text" name="age">
<p>
<input type="button" value="submit" onclick="response(this.form)">
</form>
</body>
</html>
```

www



Server-side scripting with PHP



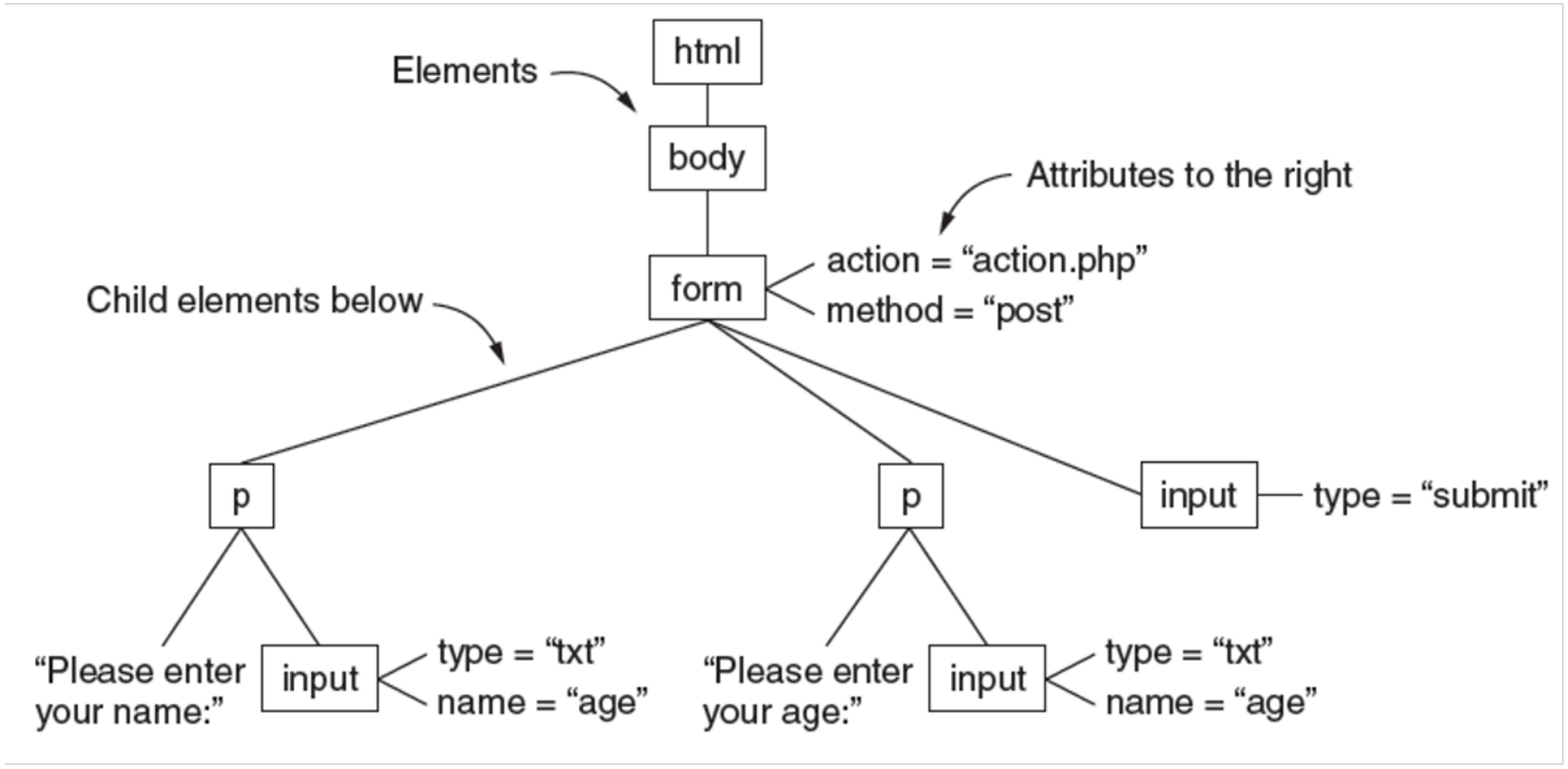
Client-side scripting with JavaScript

www

- Web applications use a set of technologies that work together, e.g. AJAX:
  - HTML: present information as pages.
  - DOM: change parts of pages while they are viewed.
  - XML: let programs exchange data with the server.
  - Asynchronous way to send and retrieve XML data.
  - JavaScript as a language to bind all this together.

WWW

- Document Object Model (DOM) represents web-pages as a tree that can be altered



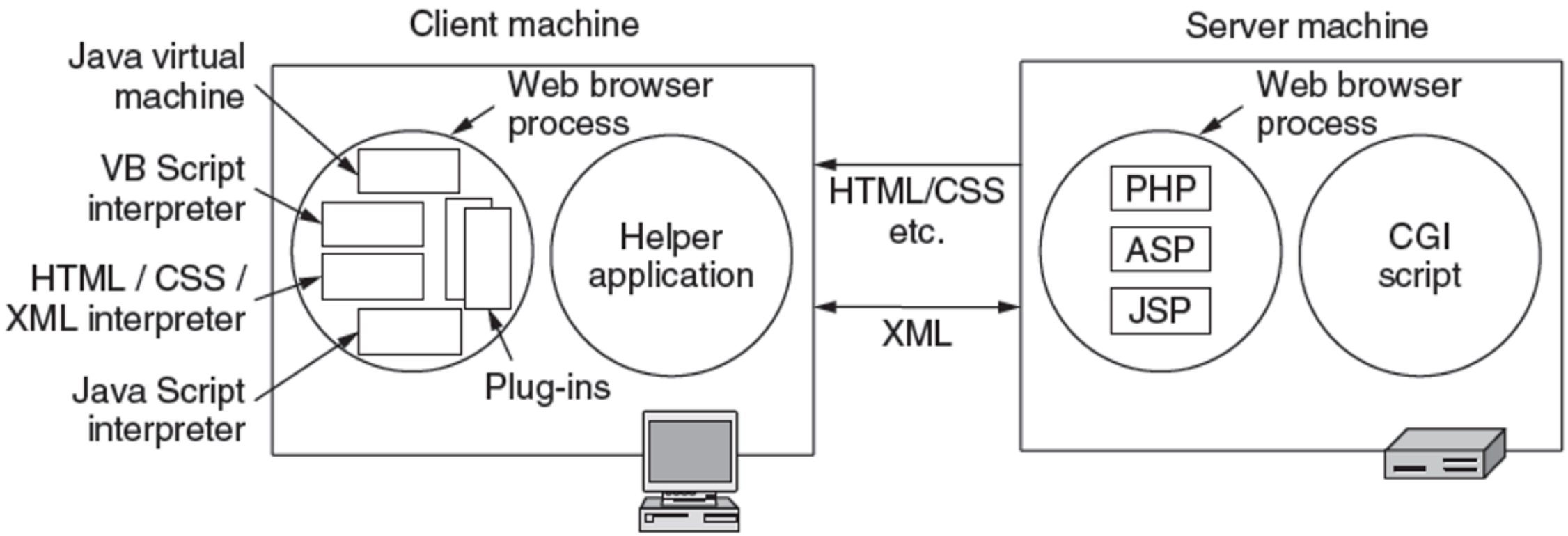
WWW

- XML captures structure

```
<?xml version="1.0" ?>
<book_list>
<book>
   <title> Human Behavior and the Principle of Least Effort </title>
   <author> George Zipf </author>
   <year> 1949 </year>
</book>

<book>
   <title> The Mathematical Theory of Communication </title>
   <author> Claude E. Shannon </author>
   <author> Warren Weaver </author>
   <year> 1949 </year>
</book>

<book>
   <title> Nineteen Eighty-Four </title>
   <author> George Orwell </author>
   <year> 1949 </year>
</book>

</book_list>
```

WWW

- Browsers and servers use sets of technology

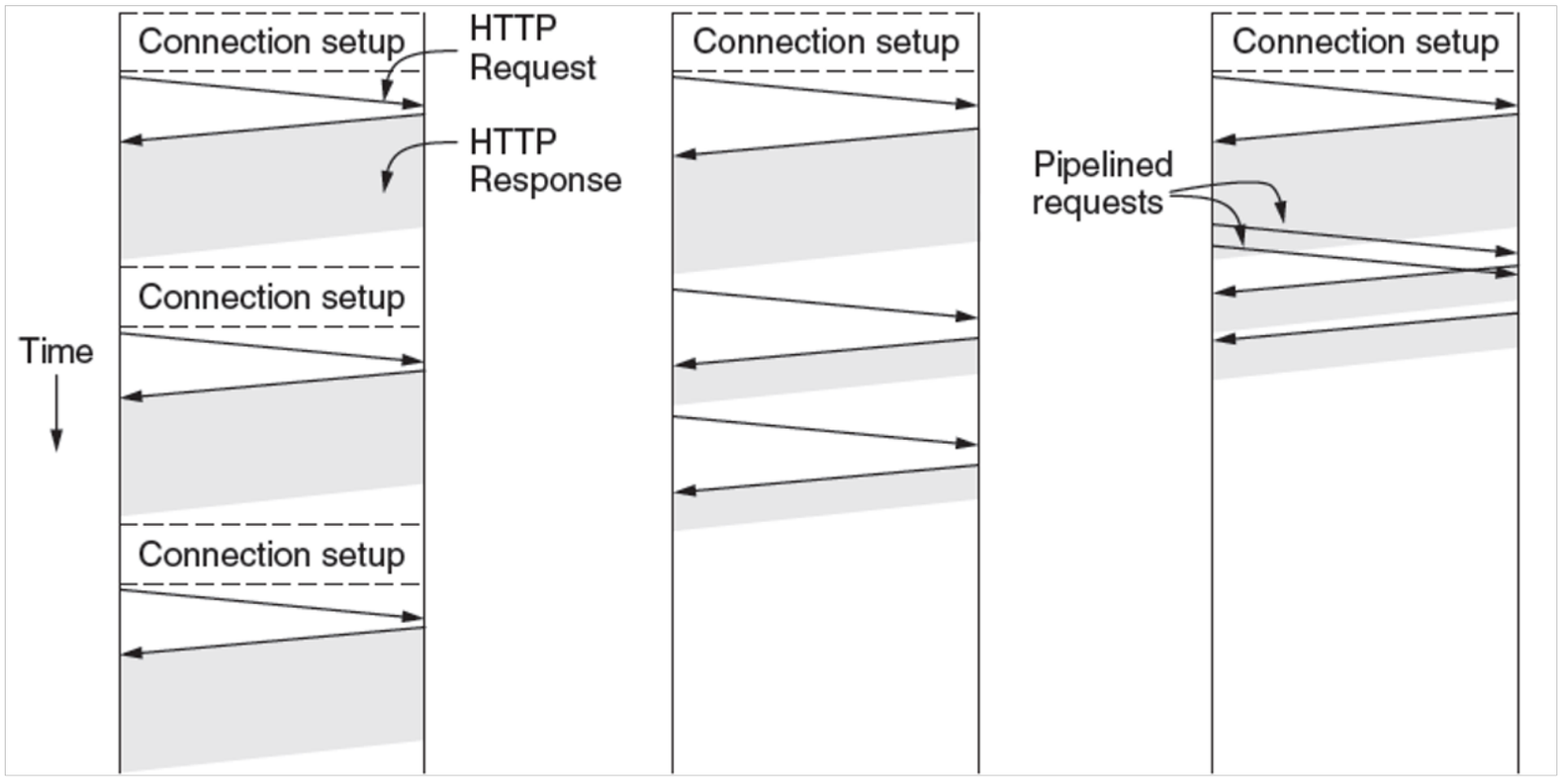


WWW

- HTTP (HyperText Transfer Protocol) is a request-response protocol that runs on top of TCP
  - Fetches pages from server to client
  - Server usually runs on port 80
  - Headers are given in readable ASCII
  - Content is described with MIME types
  - Protocol has support for pipelining requests
  - Protocol has support for caching

www

- HTTP1 uses persistent connections



One connection for each request

Sequential requests on one connection

Pipelined requests on one connection

WWW

- HTTP request methods

Fetch a page →

Used to send input data to a server program →

Method	Description
GET	Read a Web page
HEAD	Read a Web page's header
POST	Append to a Web page
PUT	Store a Web page
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Connect through a proxy
OPTIONS	Query options for a page

www

- HTTP uses response codes

Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

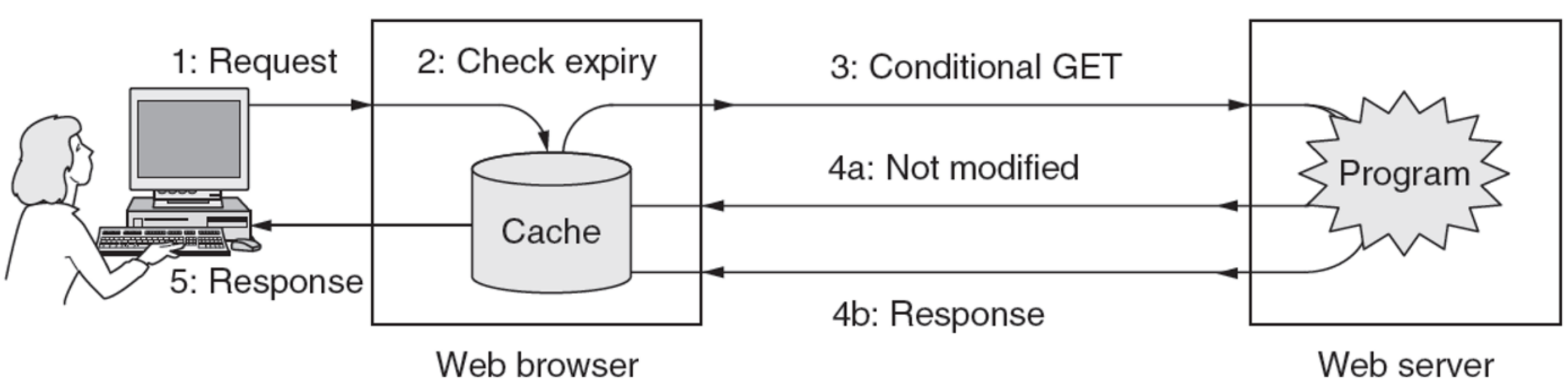
www

- HTTP headers carry information

Function	Example Headers
Browser capabilities (client → server)	User-Agent, Accept, Accept-Charset, Accept-Encoding, Accept-Language
Caching related (mixed directions)	If-Modified-Since, If-None-Match, Date, Last-Modified, Expires, Cache-Control, ETag
Browser context (client → server)	Cookie, Referer, Authorization, Host
Content delivery (server → client)	Content-Encoding, Content-Length, Content-Type, Content-Language, Content-Range, Set-Cookie

WWW

- HTTP caching checks to see if the browser has a known fresh copy, and if not if the server has updated the page
  - Uses a collection of headers for the checks
  - Can include further levels of caching (e.g., proxy)

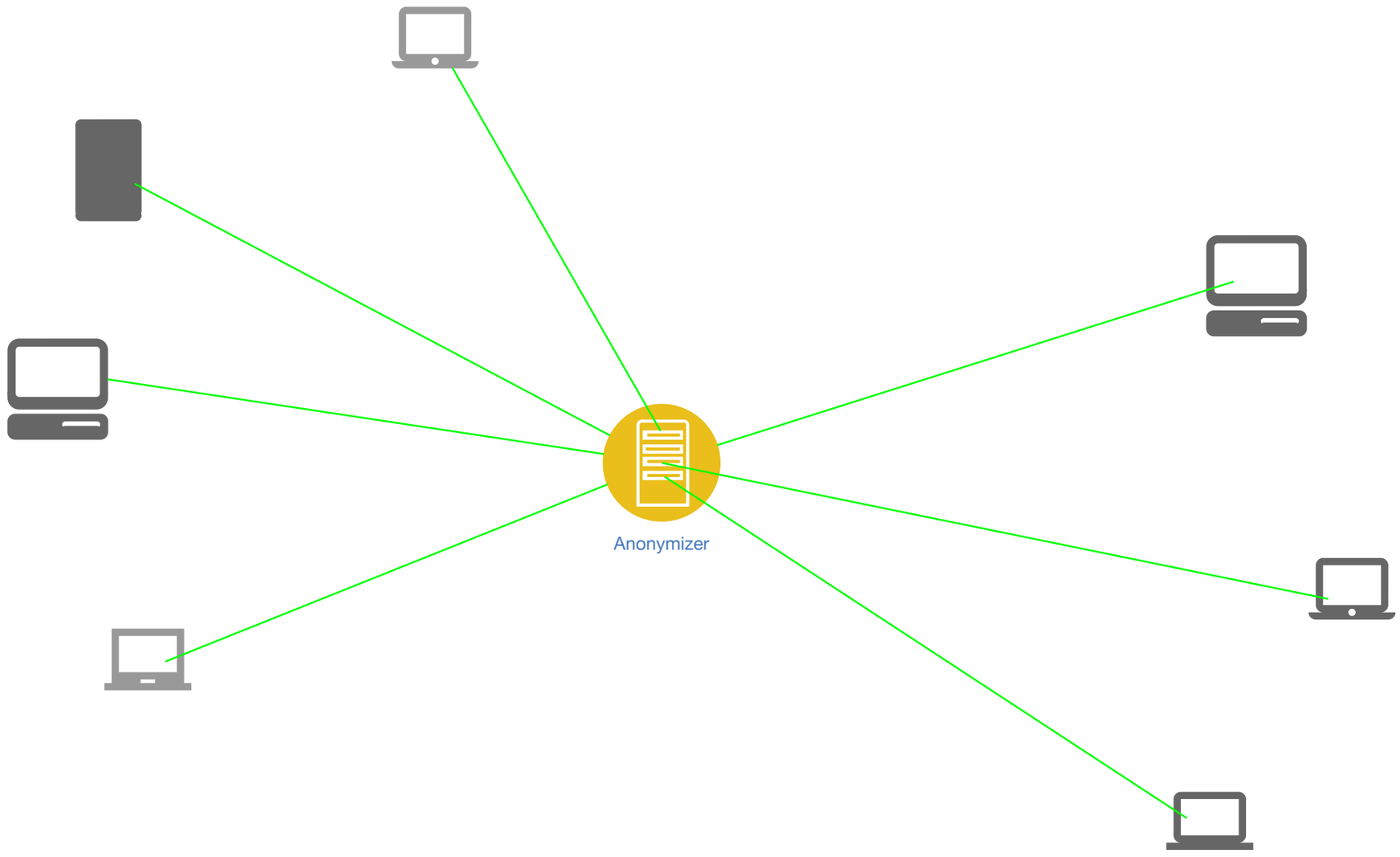


www

- Web Searches
  - Web crawling:
    - Follow links to create an index of reachable pages
    - Exception: “Deep Web”
    - Not to be confused with “Dark Web”
- Web scraping:
  - Extract information programmatically from web sites
  - `https://tschwarz.mscs.mu.edu/Classes/PDS2021/Week9Exercises/presentation.mp4`

Excursus: Dark Web

- Anonymizers
  - Use cryptography and a proxy



Excursus: Dark Web

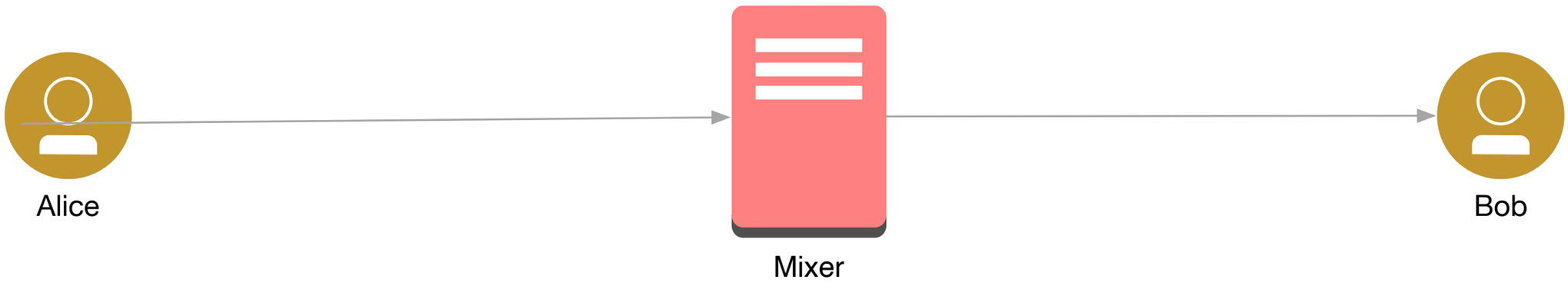
- Adversary who observes:
  - Can tell that there is traffic: Alice is communicating
  - Cannot follow a message: But with whom?
    - Anonymizers mix messages from different streams in an unpredictable manner
    - Anonymizers will generate synthetic load if necessary
- Adversary who controls the anonymizer
  - Sees all messages and traffic: Alice is telling this to Bob

Excursus: Dark Web

- Onion Routing
  - Onion routers form an overlay network
    - Communication between onion routers is encrypted
    - Chaun Mix: Store and forward device
      - Accepts fixed length messages
      - Performs cryptographic transformation
      - Forwards message to next destination in random order

Excursus: Dark Web

- Chaun Mixer with Public Keys

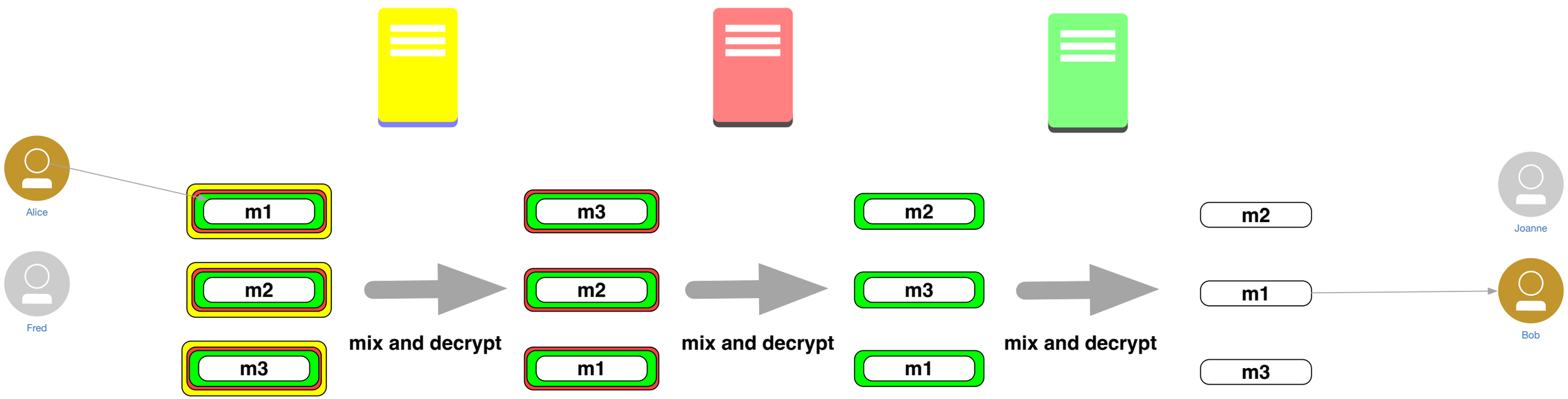


$\{r_1, \{r_0, \texttt{Message}\}_{\texttt{PK-Bob}}, \texttt{Bob}\}_{\texttt{PK-Mixer}}$

$\{r_0, \texttt{Message}\}_{\texttt{PK-Bob}}, \texttt{Bob}$

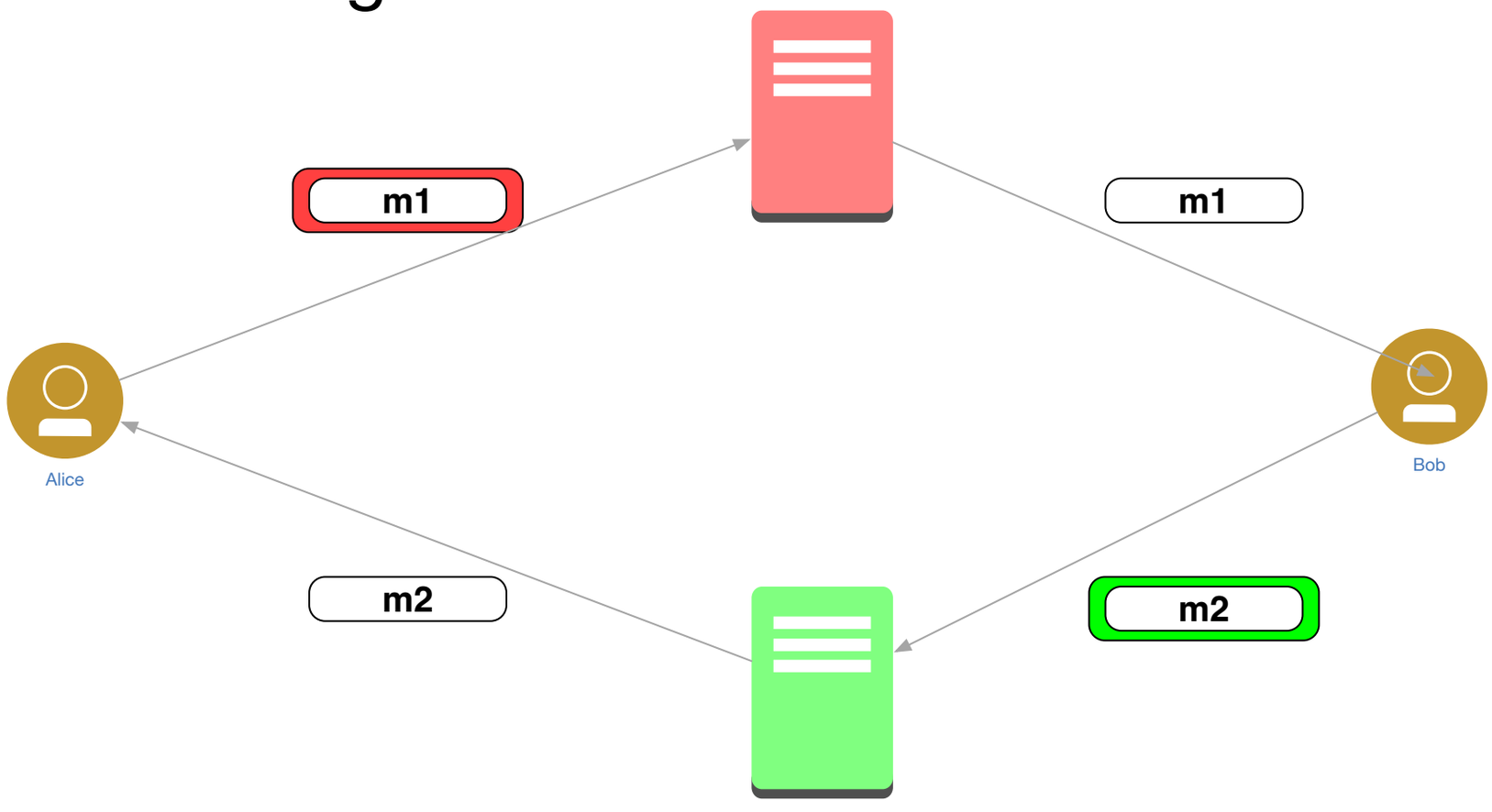
Excursus: Dark Web

- Even one Honest Chaun Mixer preserves privacy in the channel



Excursus: Dark Web

- Anonymous Return Address:
  - Alice selects a different return mixer in her request
  - Cannot establish that Alice and Bob are communicating



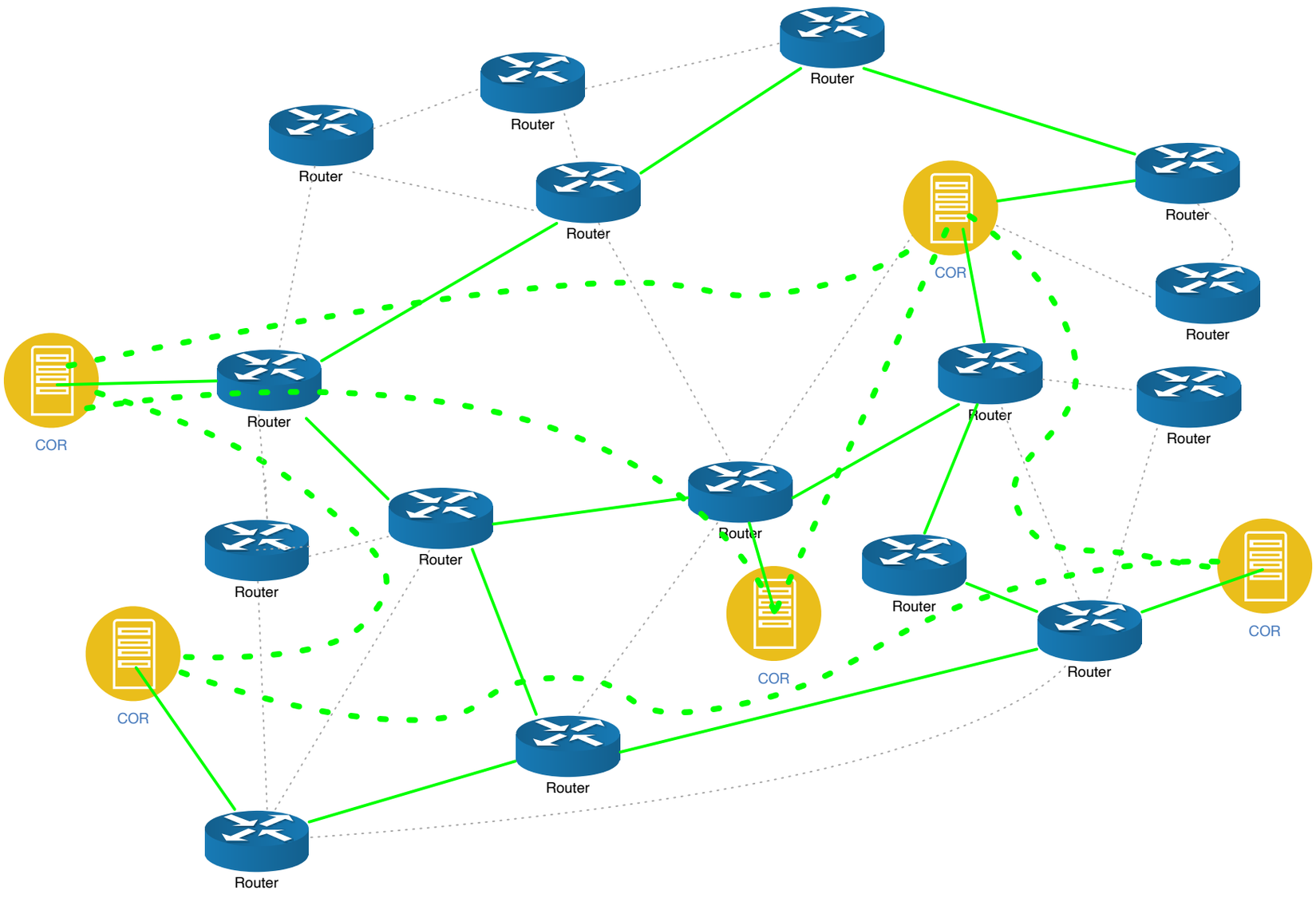
Excursus: Dark Web

- Adversary observing:
  - Can tell that Alice is communicating
  - Can tell that Bob is communicating
  - Cannot tell what Alice is saying and to whom she is talking
  - Cannot tell what Bob is saying and to whom he is talking
- Adversary that compromised a mixer:
    - At starting point: knows that Alice is communicating
    - At ending point: knows that Bob is communicating
    - In between:
      - Privacy is maintained

Excursus: Dark Web

- Onion Routing
  - Setup: Initiator creates an onion
    - Defines path through network
    - Specifies properties of the connection
      - Crypto used, Symmetric keys, …
    - Route follows TCP connections between Core Onion Routers (COR)

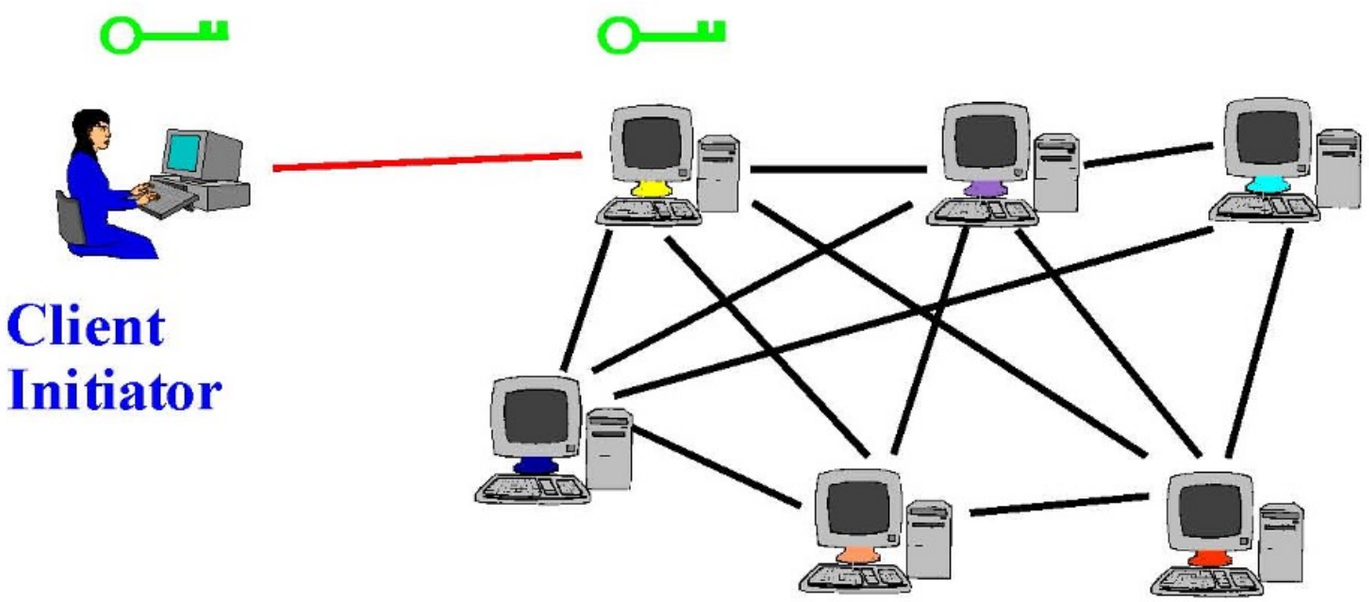
Excursus: Dark Web



COR connections form an overlay network

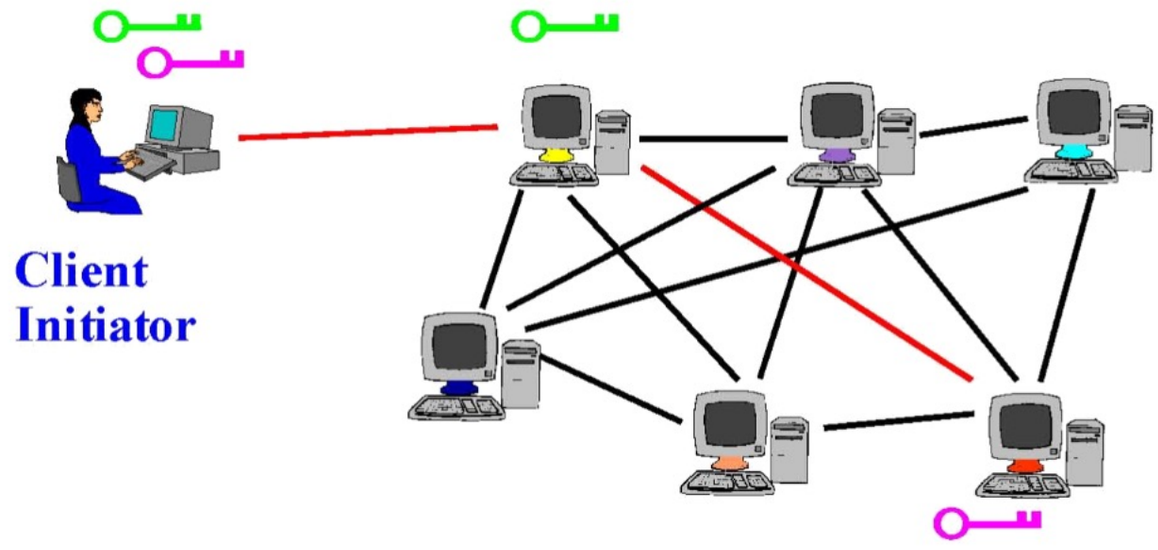
Excursus: Dark Web

- Tor Circuit Setup:
  - Client proxy establishes secret key with relay node 1



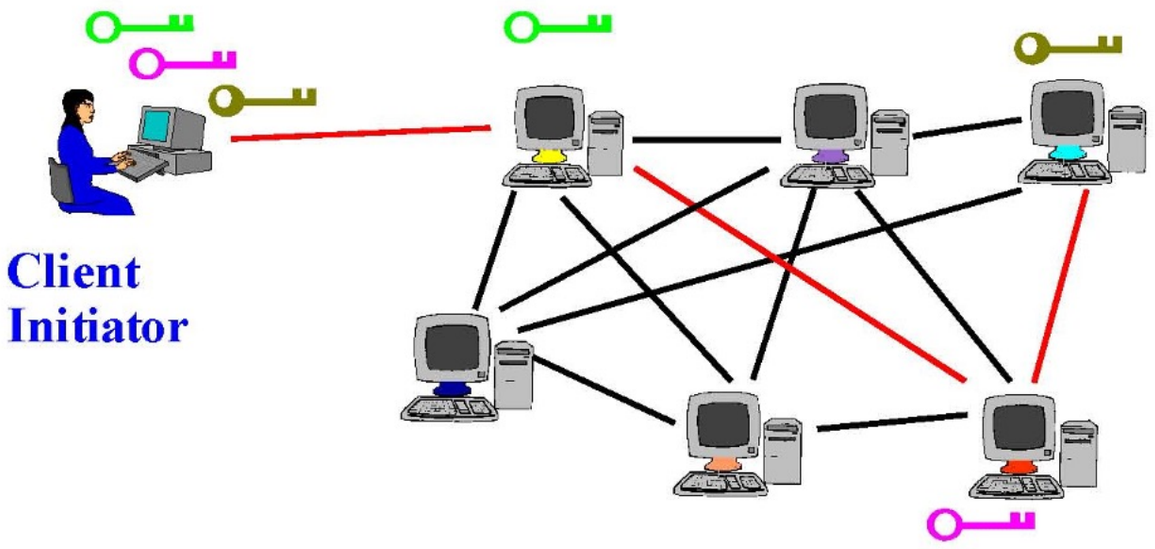
Excursus: Dark Web

- Tor Circuit Setup
  - Client proxy extends circuit by establishing a secret session key with relay node 2



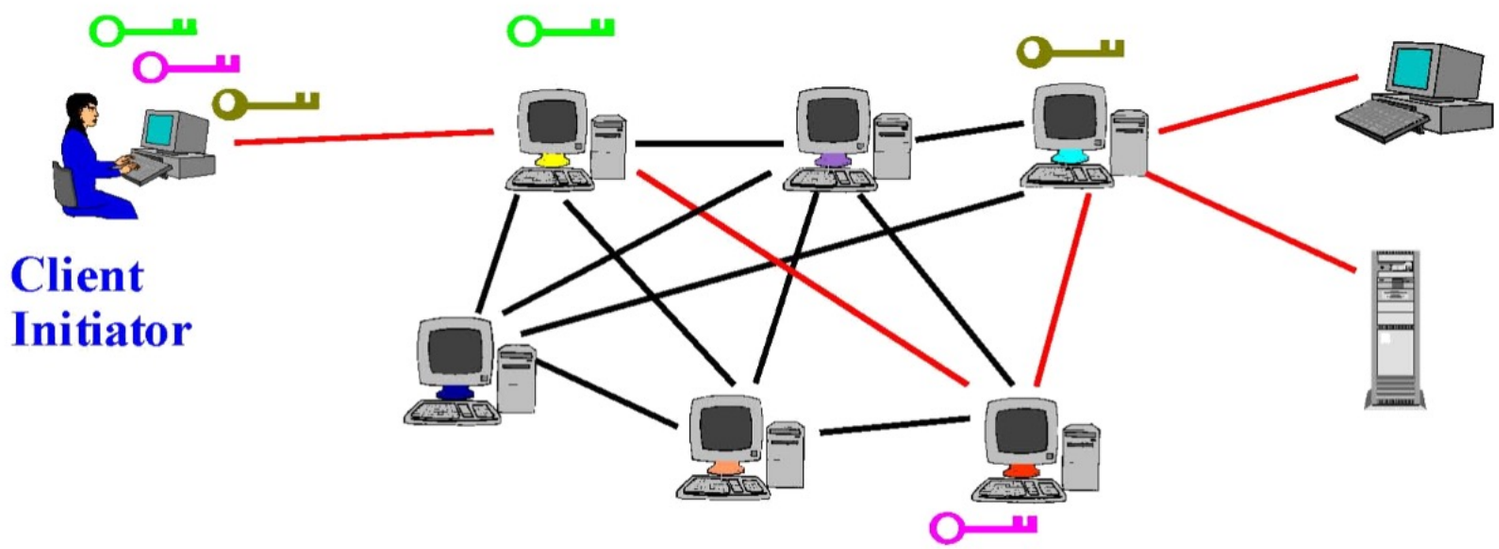
Excursus: Dark Web

- Tor Circuit Setup
  - Client proxy establishes a secret session key with relay 3



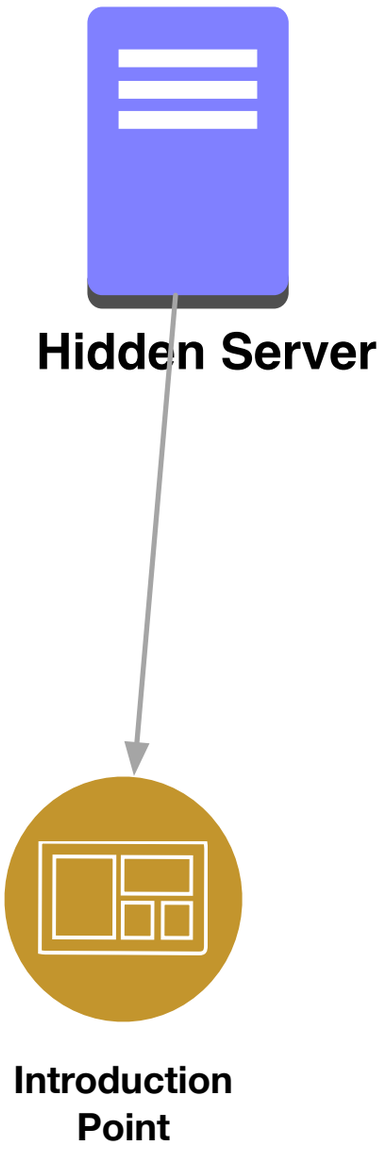
Excursus: Dark Web

- Tor Circuit Setup
  - Client application connects and communicates over the established circuit
  - Datagrams are encrypted / decrypted at each link



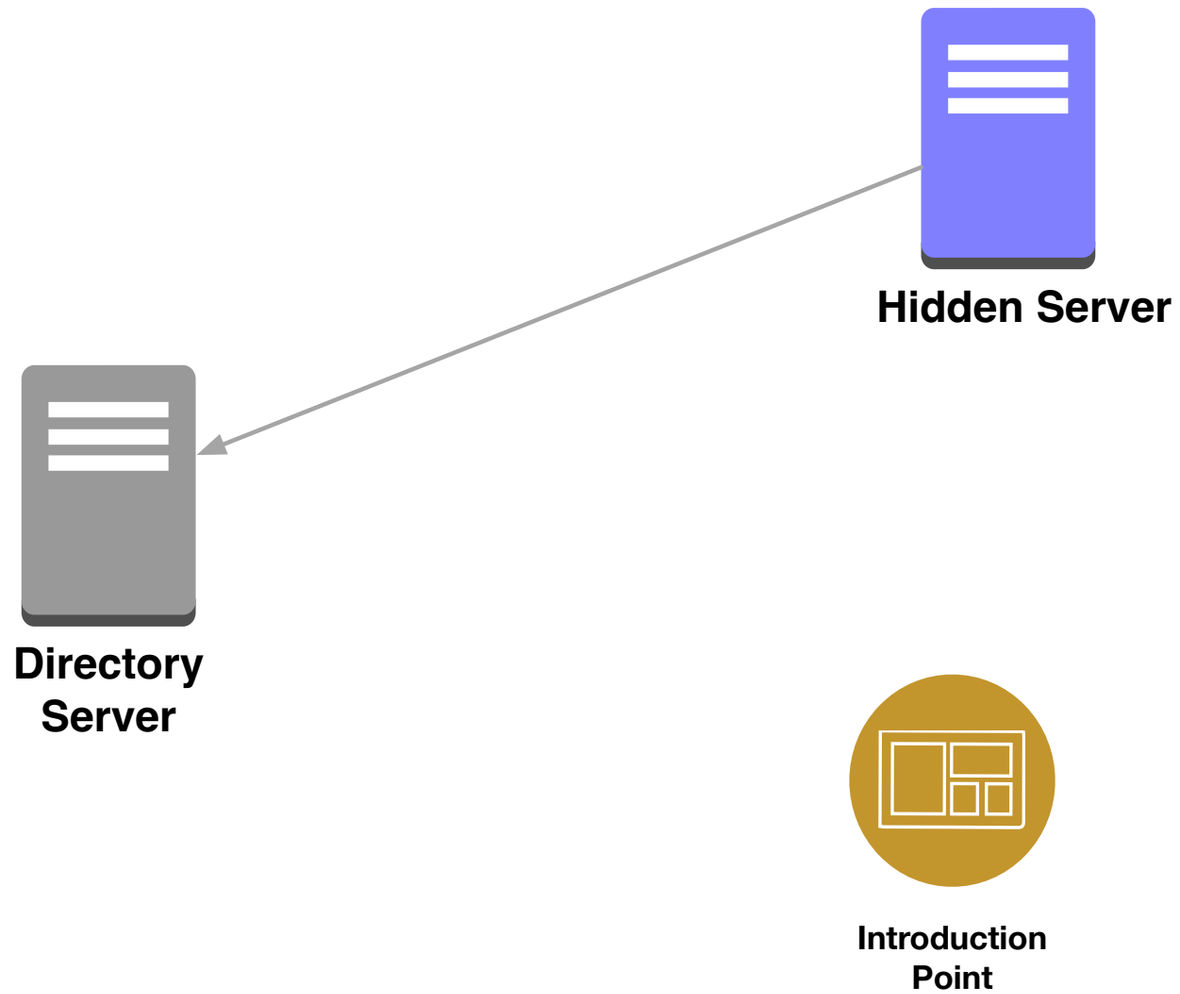
Excursus: Dark Web

- A hidden server contacts a Tor node and establishes a connection
  - Connection is kept “forever”



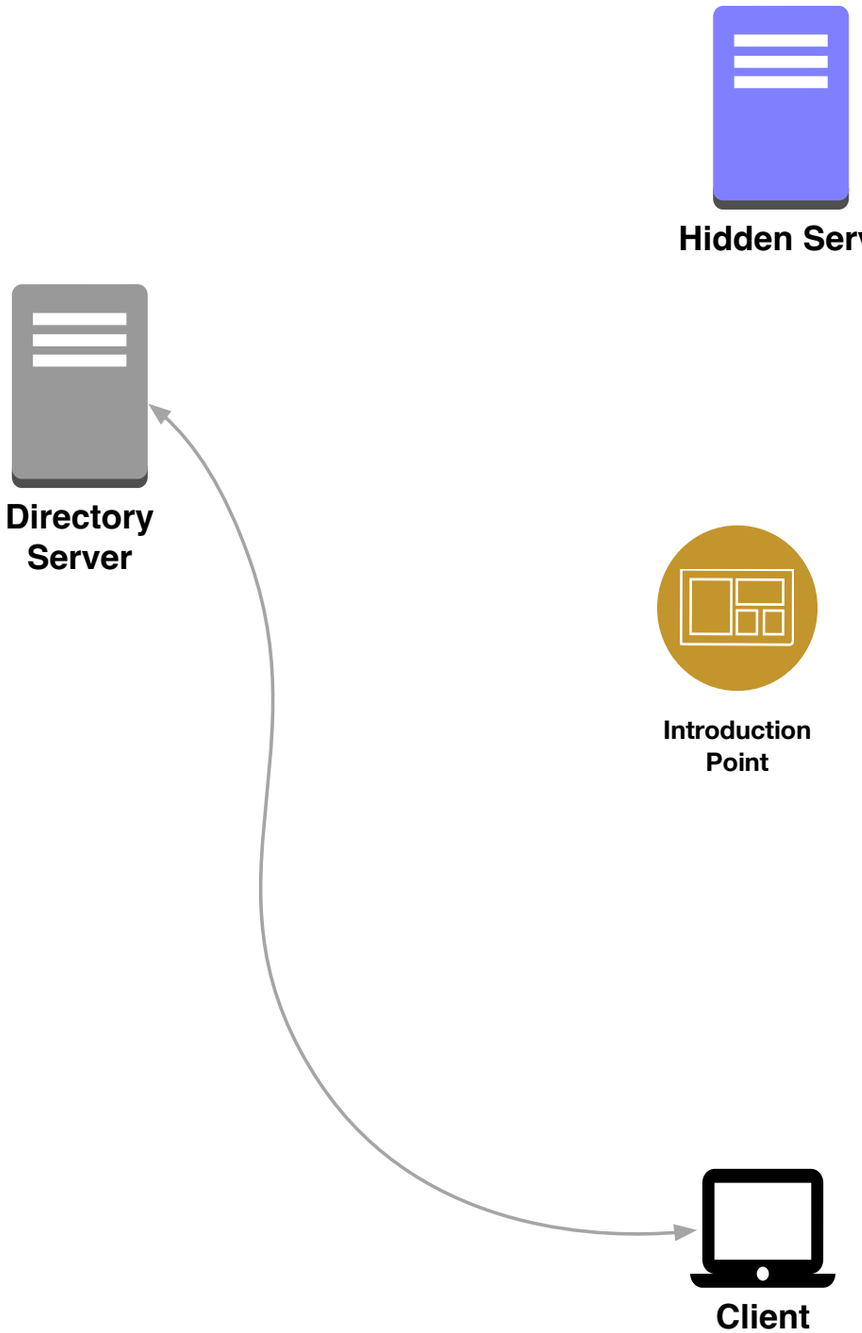
Excursus: Dark Web

- Hidden server then contacts the directory server
  - To publish contact information



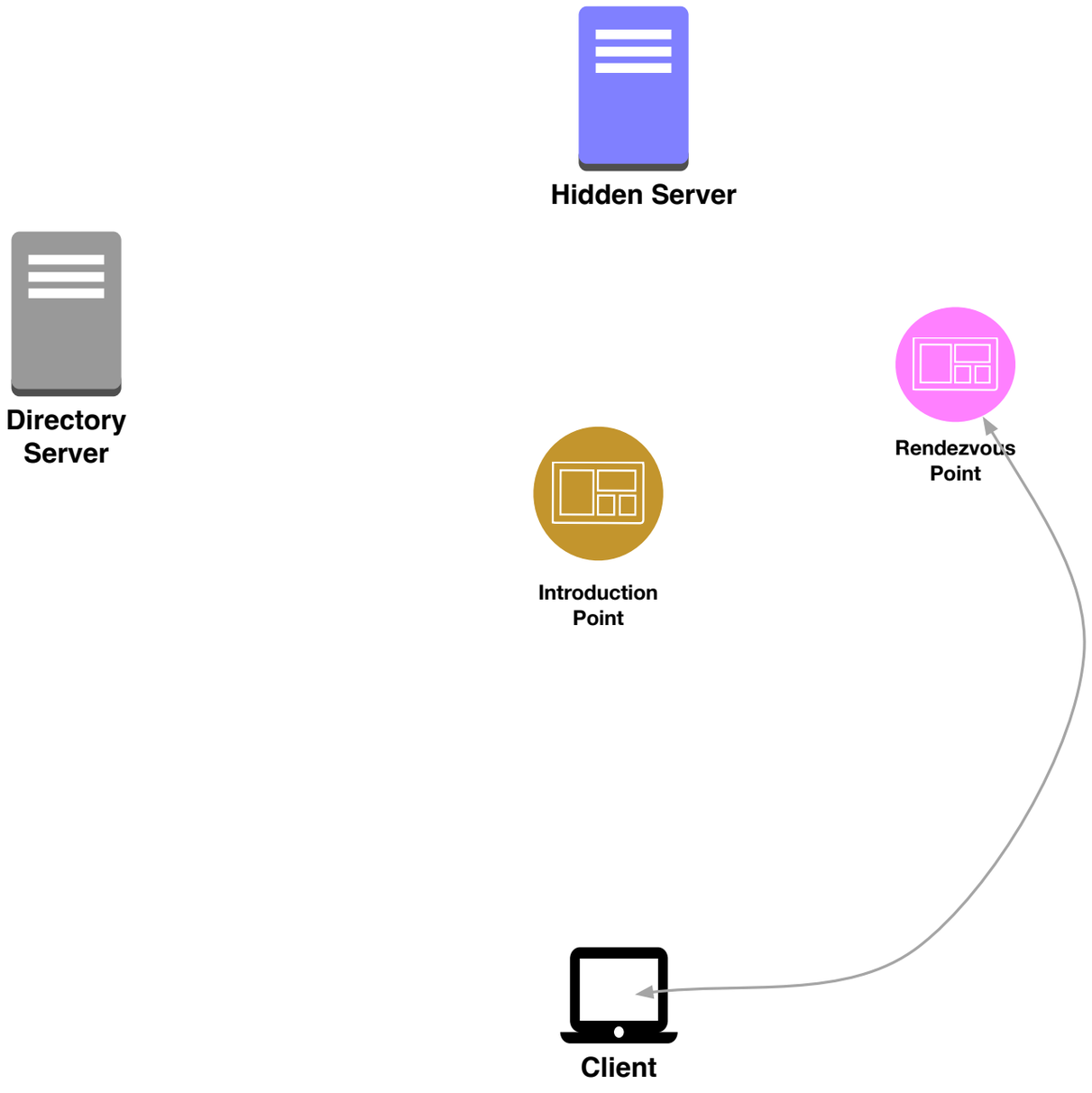
Excursus: Dark Web

- A client approaches the directory server for service
- Directory server returns contact data for hidden server
- This includes data on Introduction Point



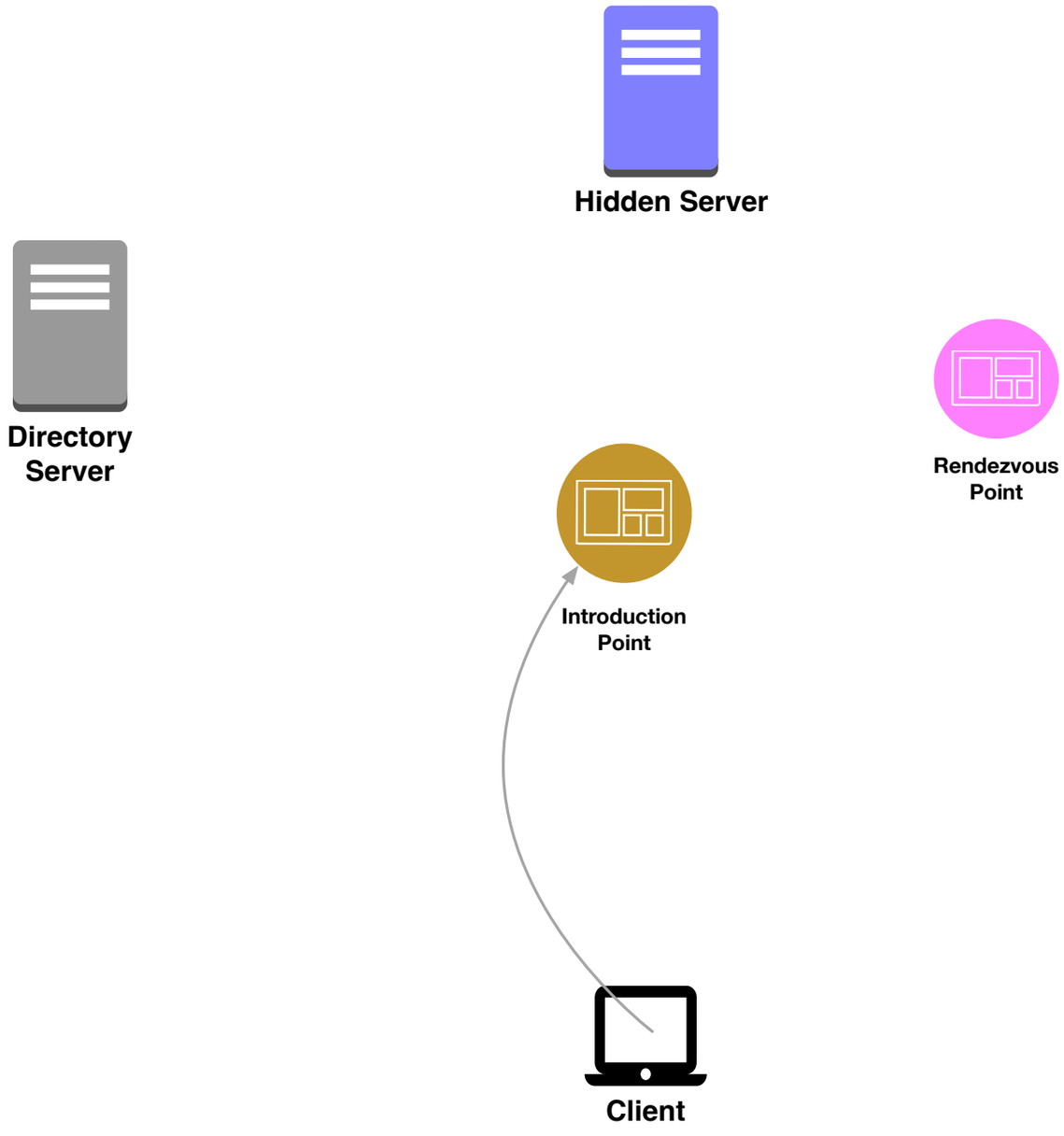
Excursus: Dark Web

- Client selects a node to act as Rendezvous Point
  - Asks it to listen for connections from a hidden service on client's behalf



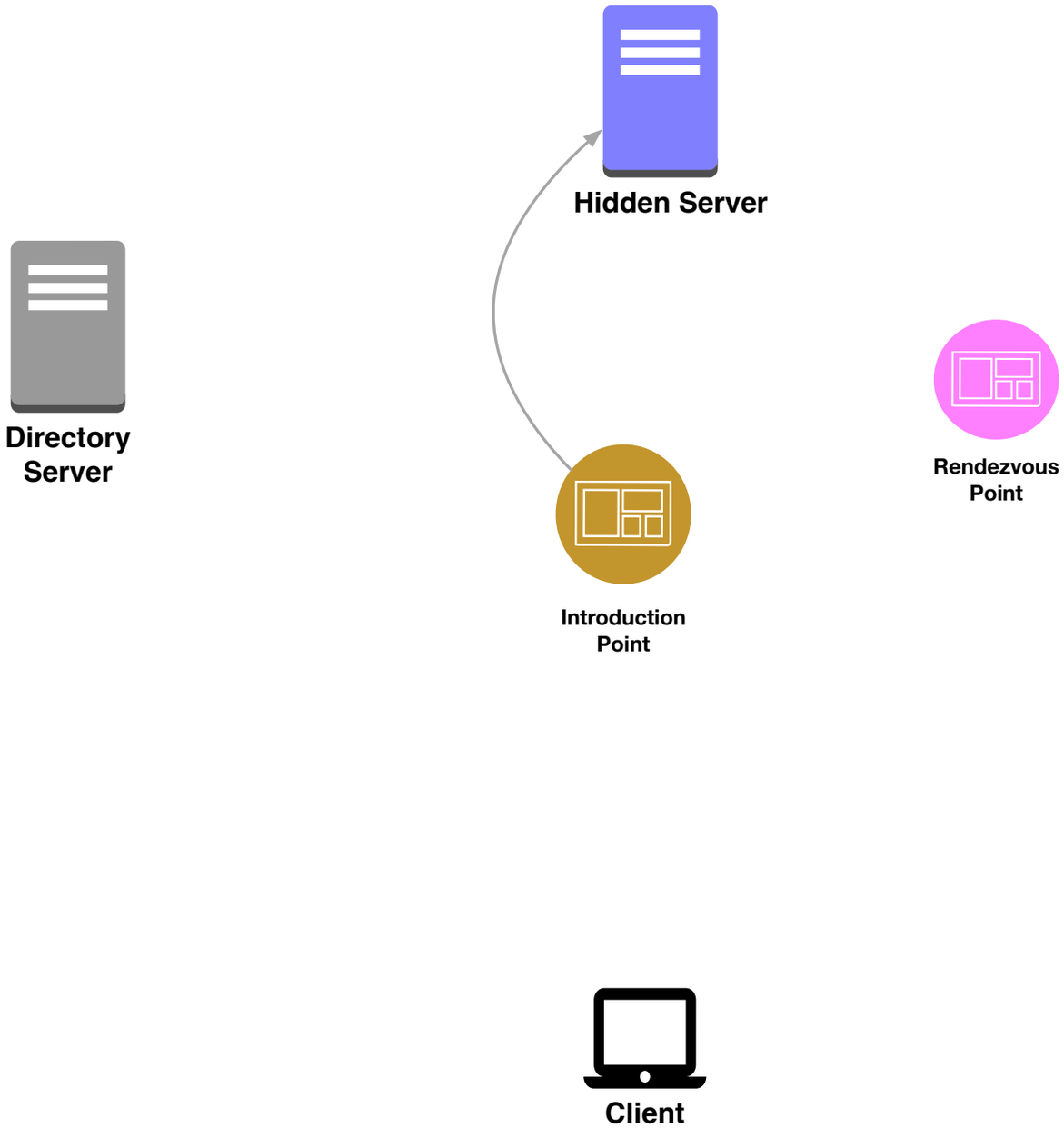
Excursus: Dark Web

- Client now contacts Introduction Point with address of Rendezvous Point



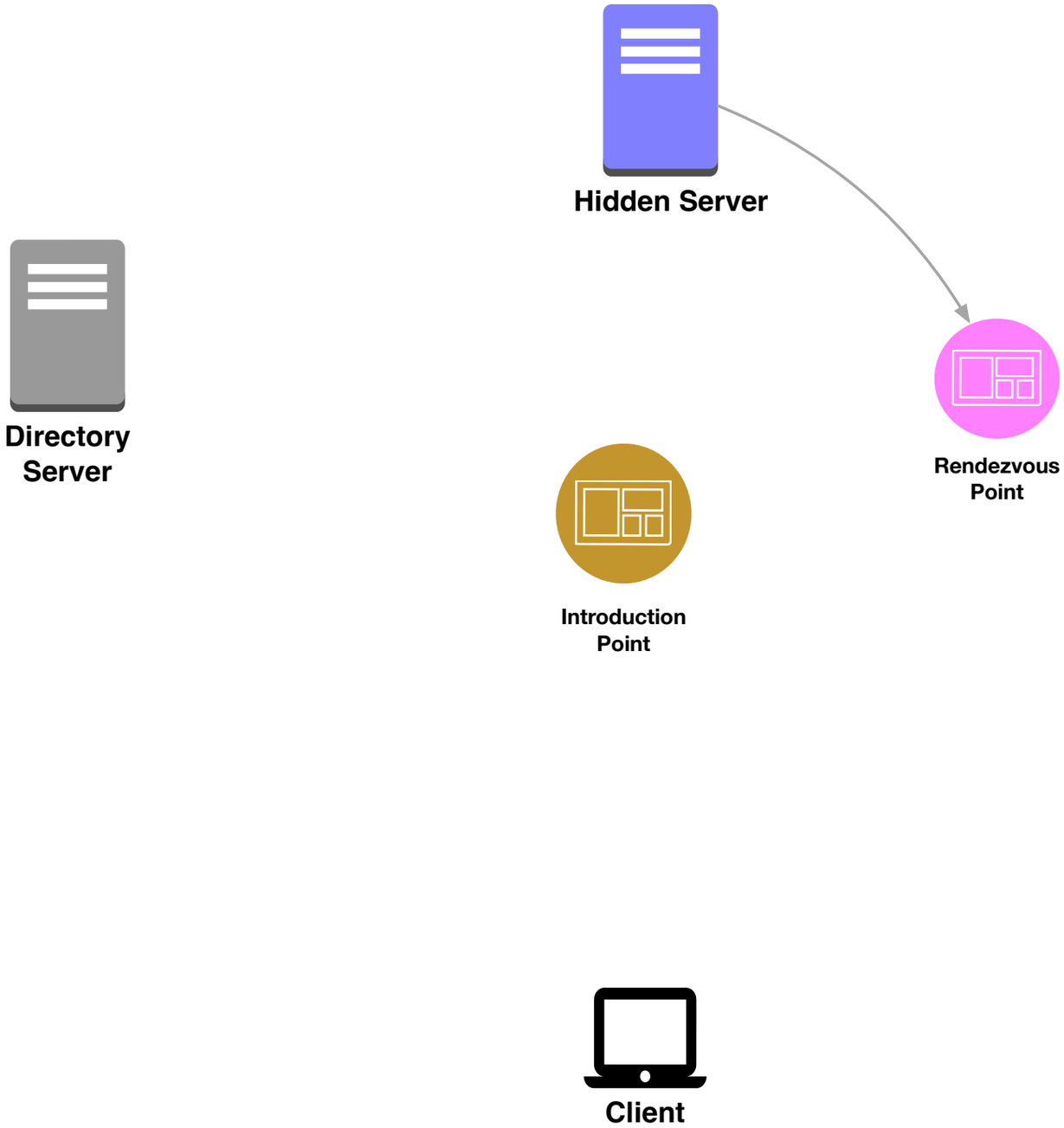
Excursus: Dark Web

- Introduction Point forwards this to the Hidden Server



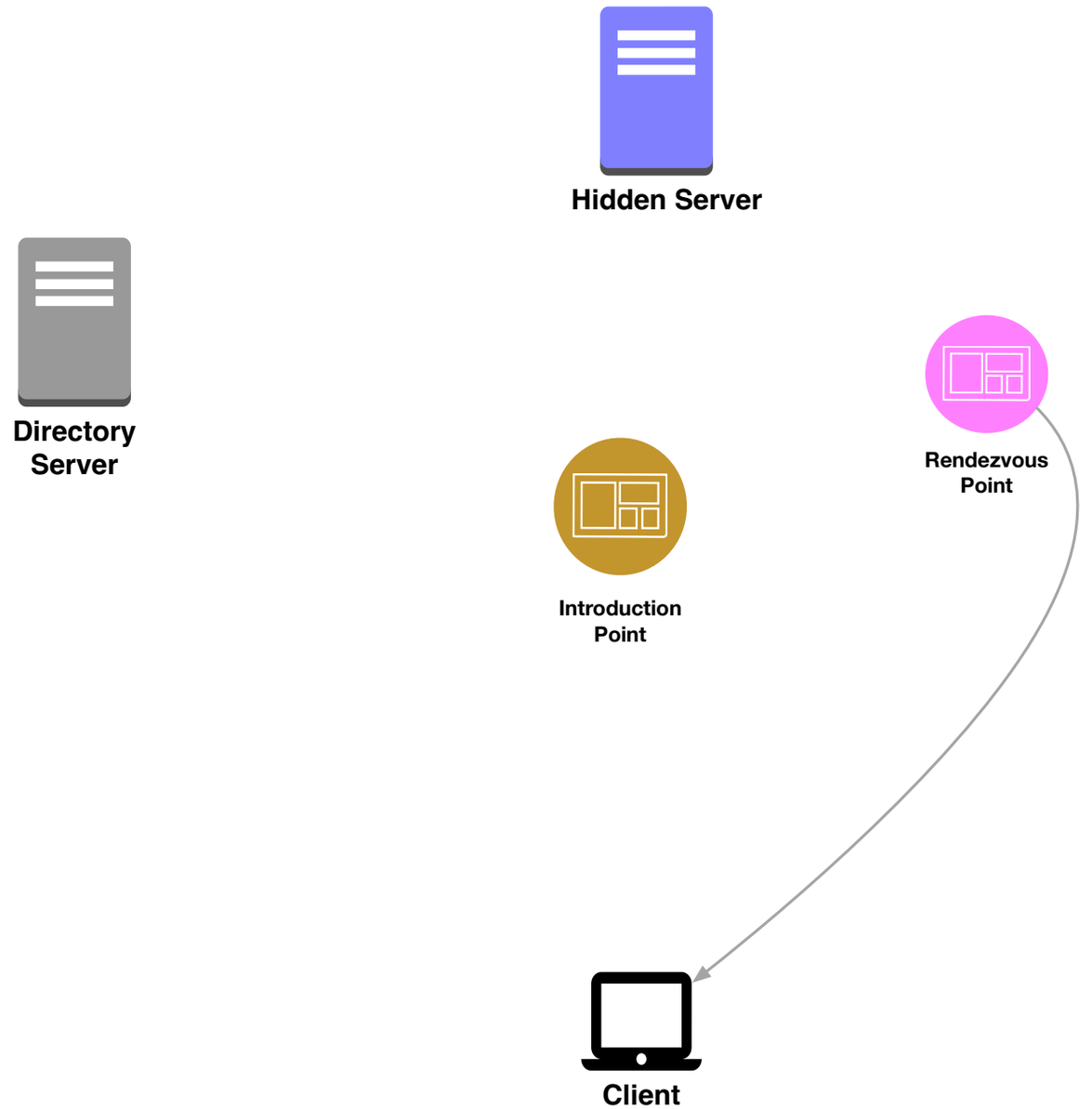
Excursus: Dark Web

- Hidden Server connects to Rendezvous Point to be connected to waiting circuit



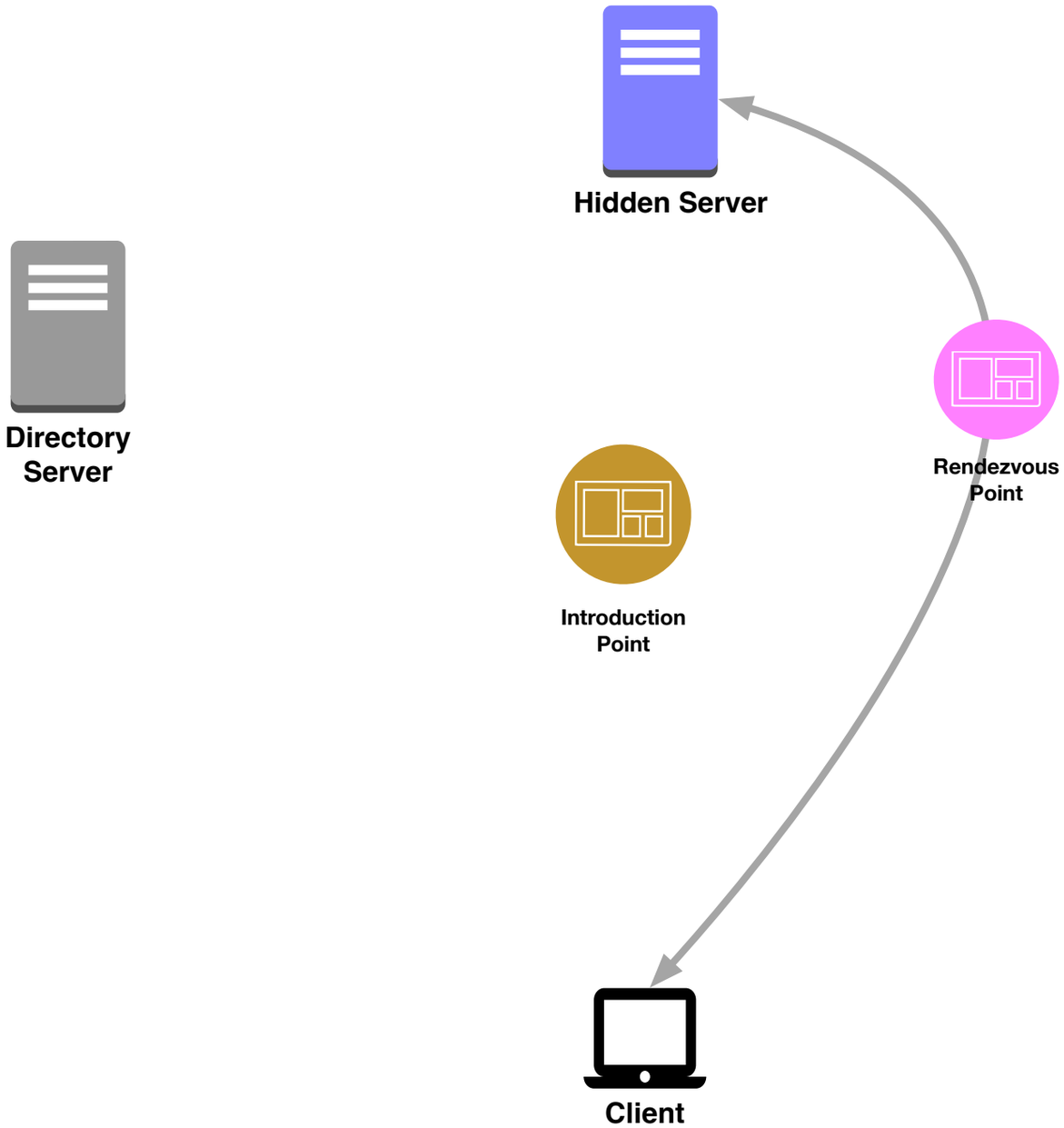
Excursus: Dark Web

- Rendezvous point reconnects to client



Excursus: Dark Web

- Hidden Server and Client have now established a Tor connection



Excursus: Dark Web

- All connections use Onion Routing
  - Client does not know the IP address of Hidden Server, but IP address of Rendezvous Point
  - Hidden Server does not know the IP address of Client, but IP address of Rendezvous Point
  - Rendezvous Point does not know the IP address of Hidden Server nor of Client
  - Tor: At least three different random nodes between Hidden Server and Rendezvous Point and two nodes between Client and Rendezvous Point

Excursus: Dark Web

- Tor does not provide absolute security
  - But it provides practical anonymity
    - For example, it is used in communication between victims of sexual assault and helping organizations
  - It is also an enabler of crime, just like bit-coin

The Web 2.0

- Shift away from class request-response, to asynchronous communication
- Web browsers request data without human intervention (e.g., without clicking on link/button)
- Such requests can mask latency or limited bandwidth
  - pre-fetching
  - displaying before all data has arrived
- HTTP requests are becoming automated, rather than human generated
- AJAX is one such supporting technology

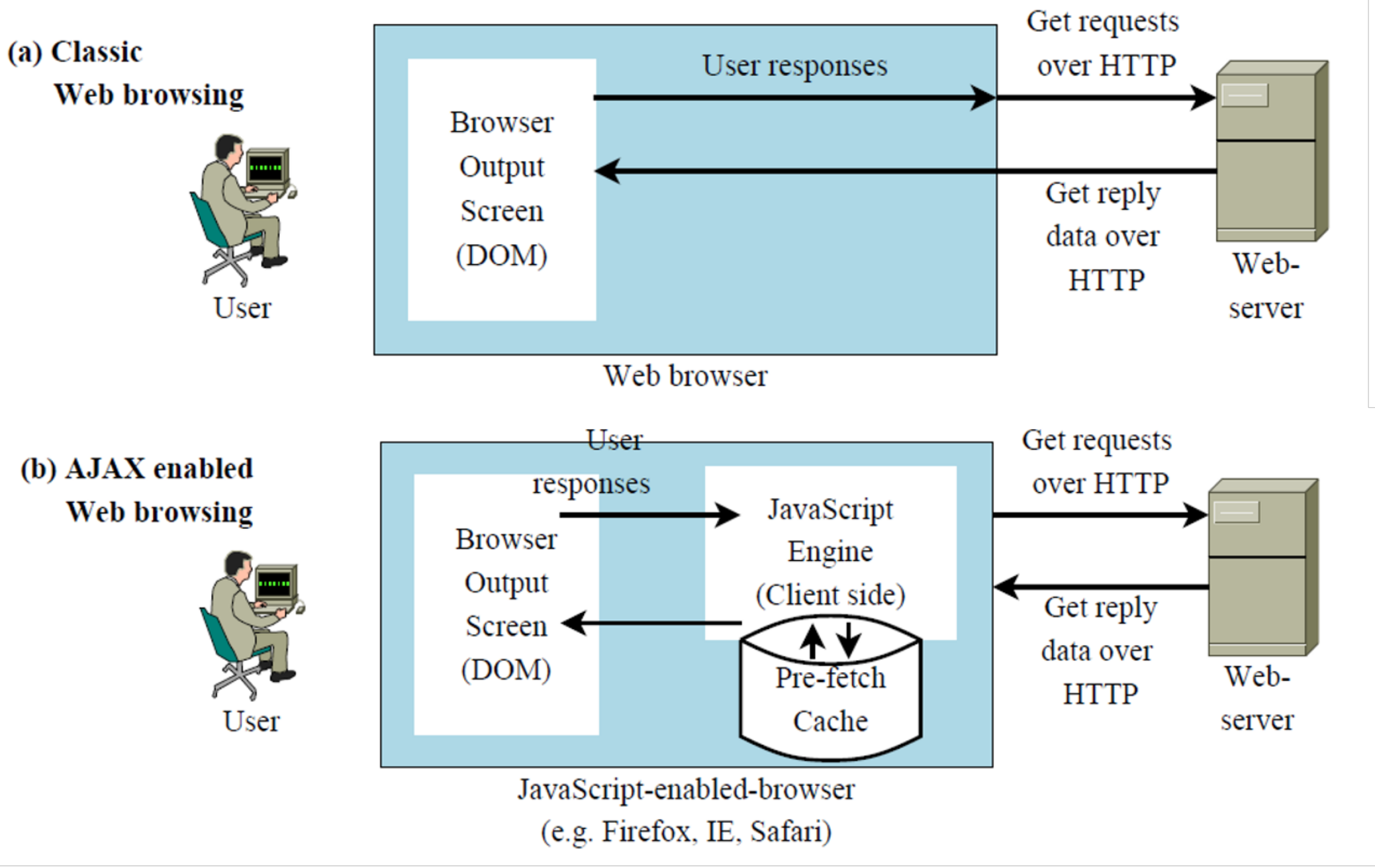
The Web 2.0

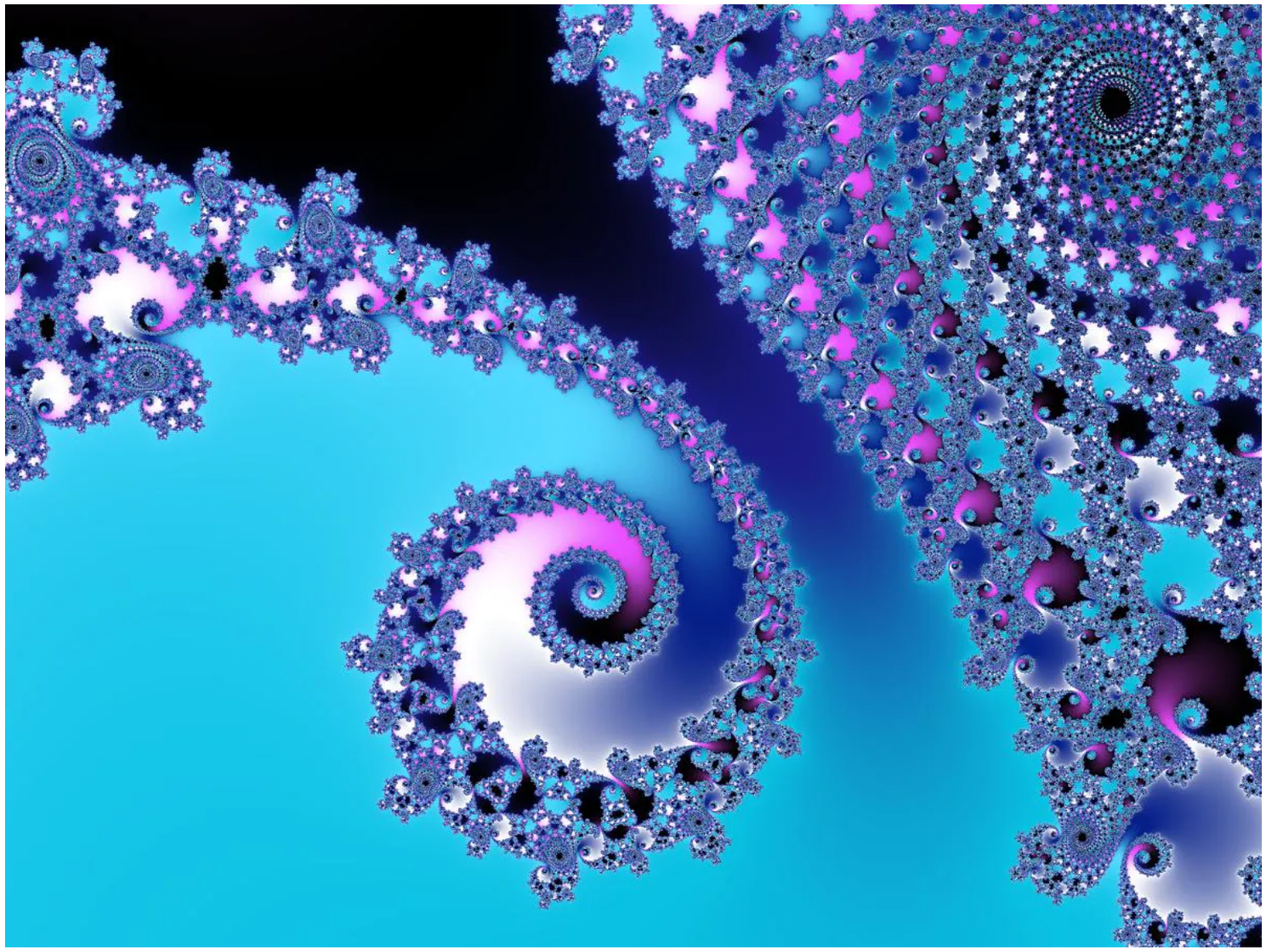
- AJAX – acronym for Asynchronous JavaScript and XML
  - Despite name, doesn't have to be XML (often JSON, a JavaScript text-based notation)
  - Not stand-alone language or technology
  - Methods for client-side code to create Web applications
- Clients send and receive data asynchronously, without interfering with display

The Web 2.0

- AJAX uses:
  - Javascript (for altering page)
  - XML or JSON (for information exchange)
  - ASP or JSP (server side)
- Key is the XMLHttpRequest object
- All modern browsers support XMLHttpRequest object
  - Note: IE5 and IE6 uses an ActiveXObject.XMLHttpRequest object exchanges data with server asynchronously
  - Update parts of Web page, without reloading whole page

The Web 2.0

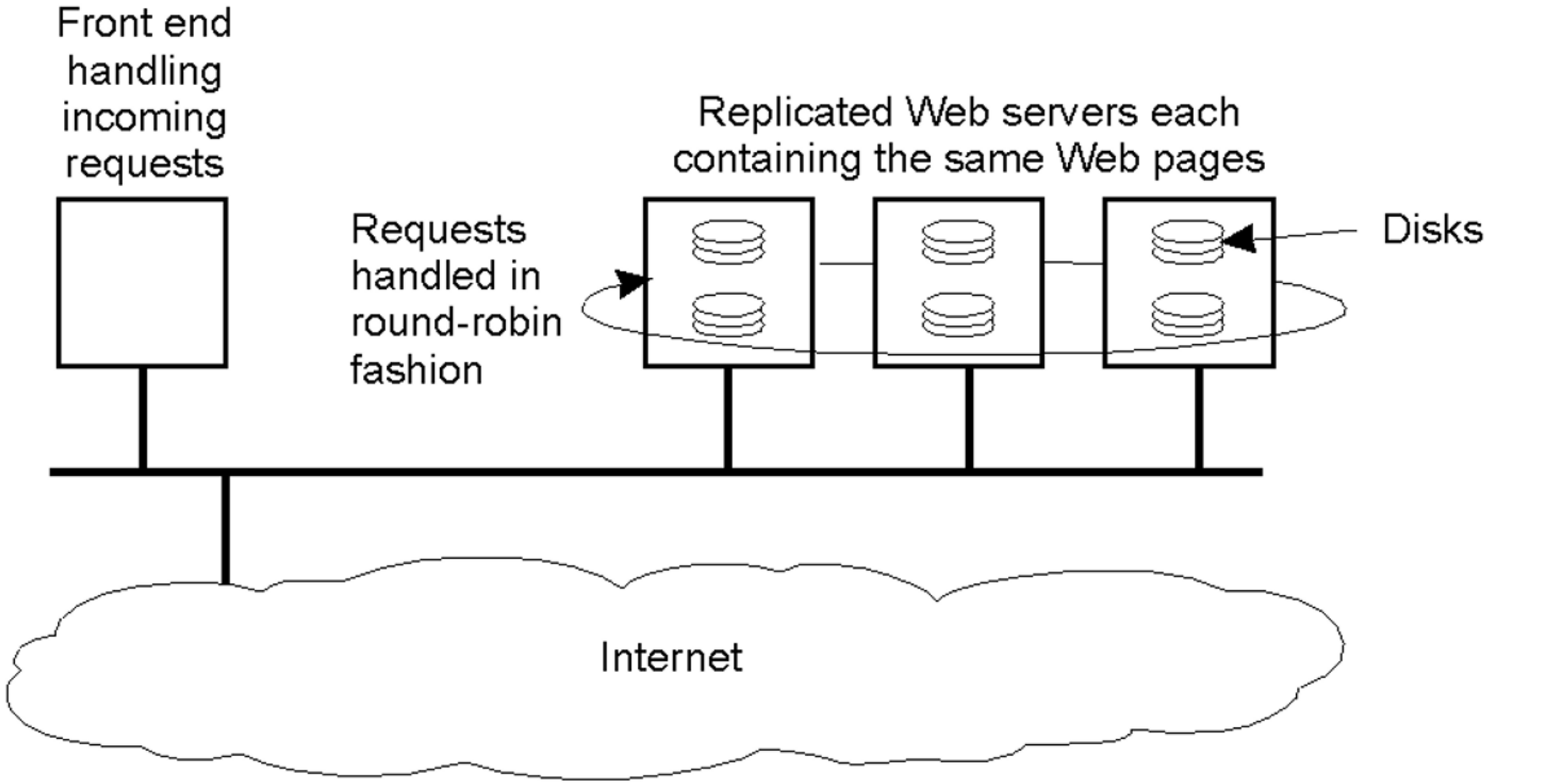

(a) Classic
Web browsing
User
Browser
Output
Screen
(DOM)
User responses
Get requests
over HTTP
Get reply
data over
HTTP
Web-
server
Web browser
(b) AJAX enabled
Web browsing
User
User
responses
Browser
Output
Screen
(DOM)
JavaScript
Engine
(Client side)
Pre-fetch
Cache
Get requests
over HTTP
Get reply
data over
HTTP
Web-
server
JavaScript-enabled-browser
(e.g. Firefox, IE, Safari)



Distributed Systems

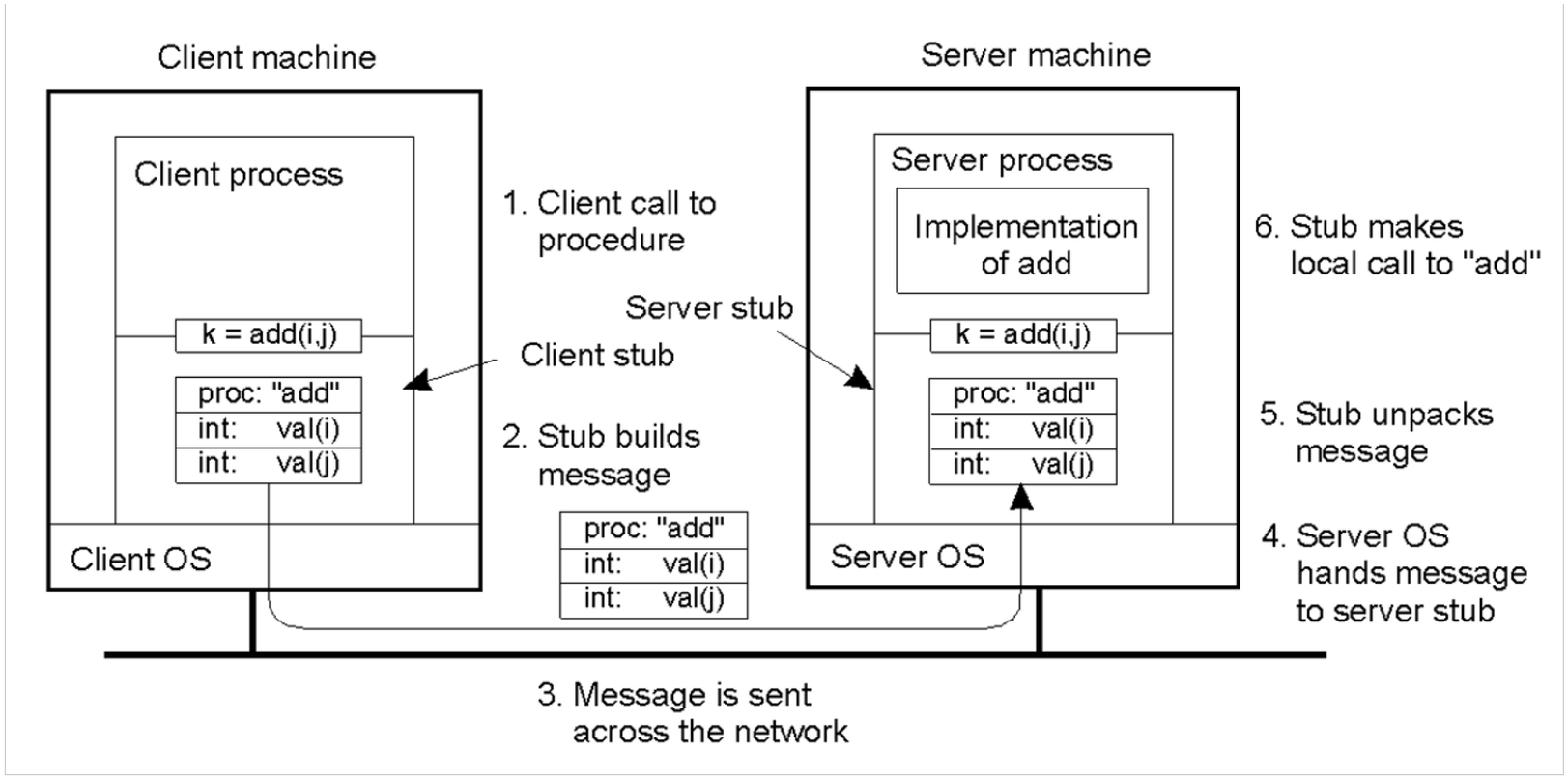
Thomas Schwarz, SJ

Distributed Systems

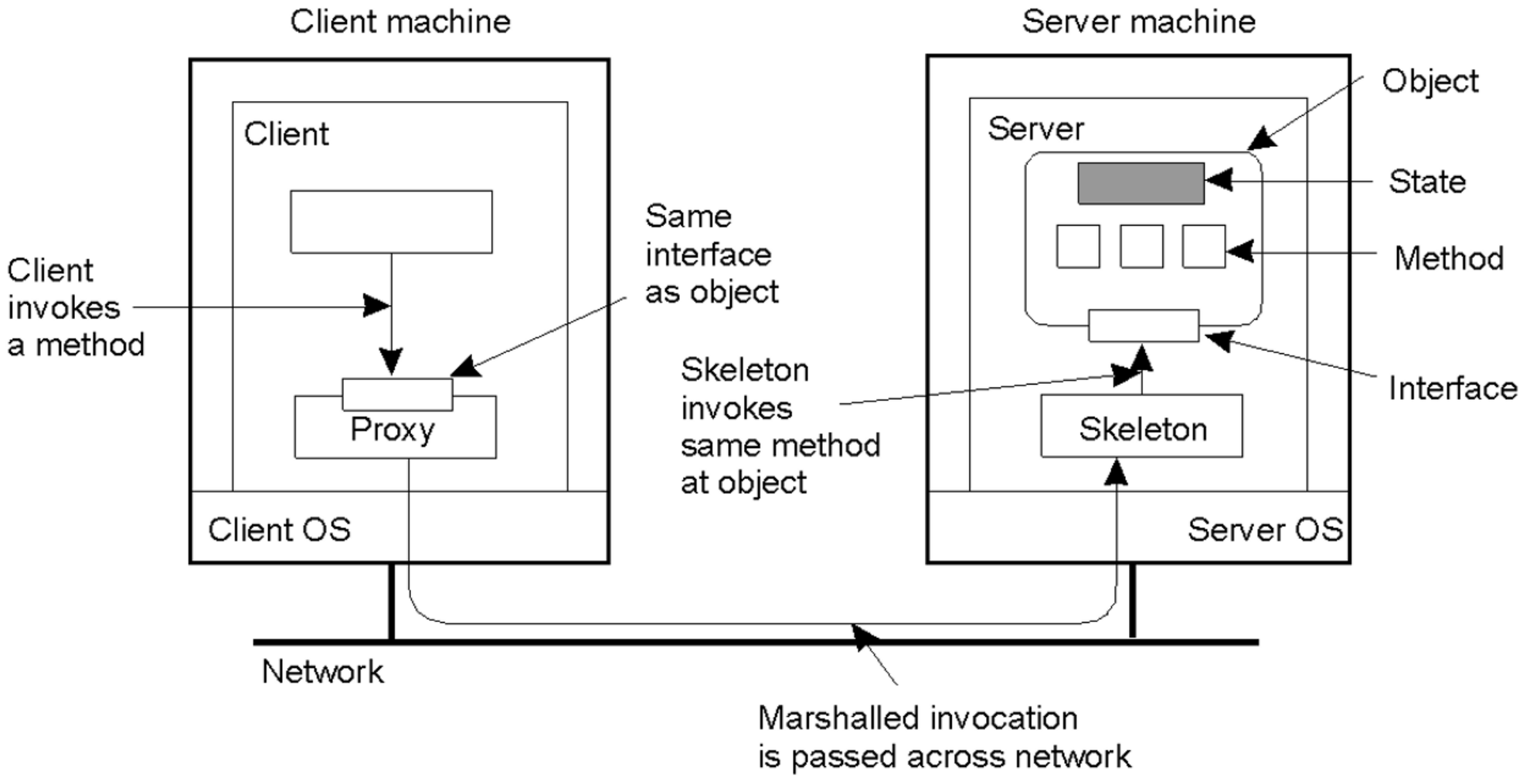


Distributed Systems

- Remote procedure calls
  - “Old Architecture”



Distributed Systems



Distributed Objects

Content Delivery Systems

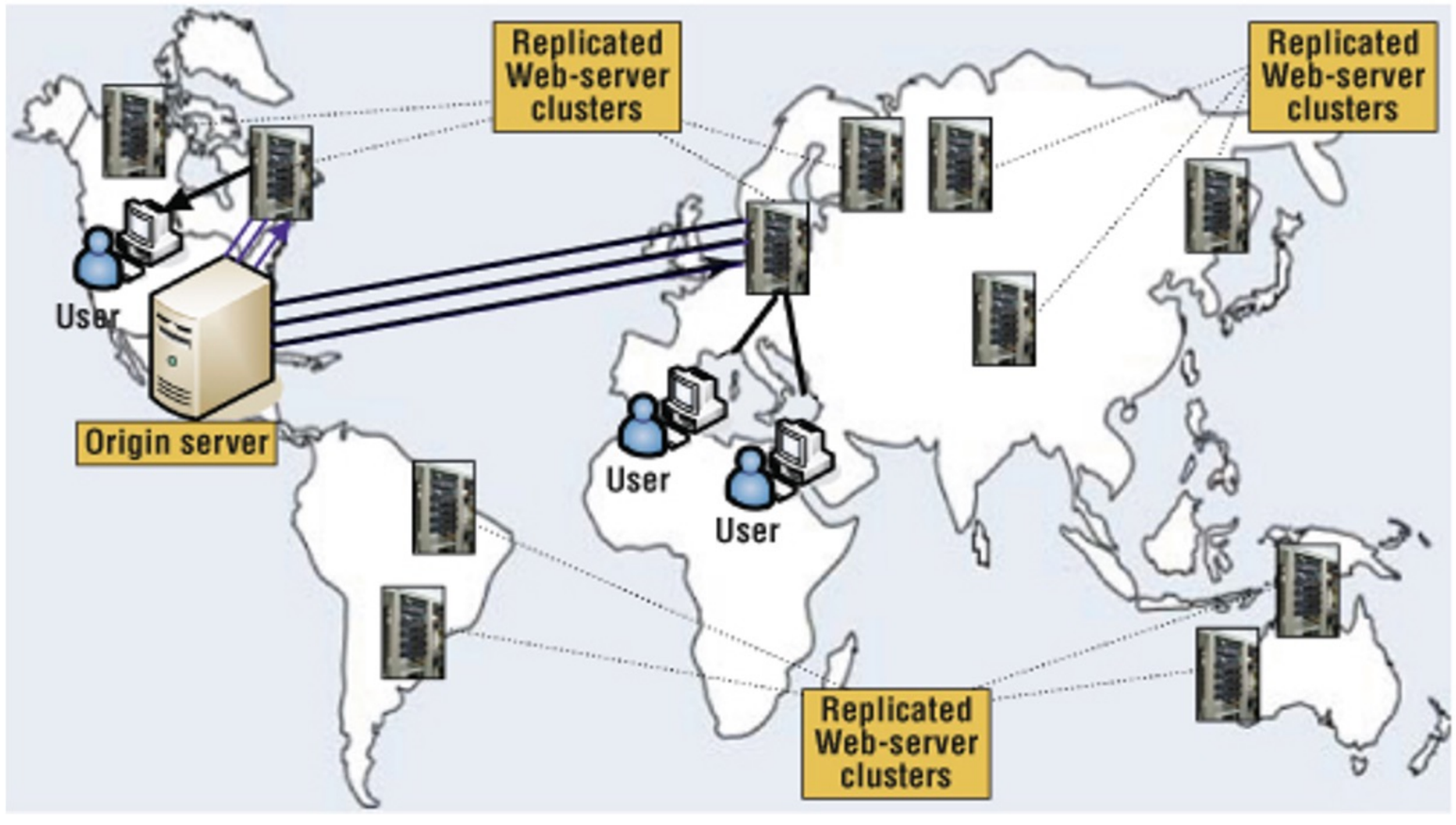
- 1998 – 1st CDNs appear. Save $ by putting more web sites on a CDN, reliability and scalability without expensive hardware and management
- 1999 – several companies (Akamai, Mirror Image) became the specialists in providing fast and reliable delivery of Web content, earning large profits
- 2000 – U.S. only, CDNs are a huge market generating $905 millions, reaching $12 billion by 2007
- 2001 – the flash crowd event (numerous users access a web site simultaneously), e.g., Sept. 11 2001 when users flooded popular news sites, making the sites unavailable. Flash events transfer more $ to CDN sale income
- 2002 – Large-scale ISPs (AT&T) tend to build their own CDN functionality, providing customized services

Content Delivery Systems

- What: Geographically distributed network of Web servers around the globe (by an individual provider, E.g. Akamai).
- Why: Improve the performance and scalability of content retrieval.
- How: Allow several content providers to replicate their content in a network of servers.

Content Delivery Systems

- Example: Akamai



Content Delivery Systems

- One of the main goals of CDNs is to put content provider in control over how her content is cached
- Content provider signs a contract with CDN
  - Contract specifies how content can be cached
  - Contract also means CDN will follow what content provider wants
- CDNs typically charge per-byte of traffic served
- CDNs can be used for any kind of content
  - Typically main use is for web content
- Streaming media has also been delivered over CDNs

Content Delivery Systems

- Original servers
- A set of surrogate servers or CDN servers
  - Geographically distributed worldwide
  - Cache original servers' content
- Routers
  - deliver the client's requests to a best fitted CDN server (latency, load balancing, etc)
- •Network elements
  - Distribute content from the original servers to surrogate/CDN servers
- Accounting mechanism
  - Provide logs and accounting info. to the original servers

Content Delivery System

- Users send requests to origin server
- Requests somehow intercepted by redirection service
- Redirection service forwards user's request to the "best" CDN content server
- Content served from the CDN content server

Content Delivery System

- DNS redirection
  - Authoritative DNS server is controlled by the CDN infrastructure. Distributes the load to the various CDN servers depending whatever policy (e.g. round-robin, least loaded CDN server, geographical distance etc.) using DNS trick.
- URL rewriting
  - Main page still comes from the origin server, but URL for the embedded objects, e.g. images, clips are rewritten, which points to a any of the CDN server. Some vendors rewrite using hostname and some uses IP address directly.

Content Delivery System

- Full redirection
  - Any request for origin server is redirected to CDN
  - Basically, CDN takes control of content provider's DNS zone
- Partial redirection
  - Content provider marks which objects are to be served from CDN
  - Typically, larger objects like images are selected
    - Refer to images as: <img src=http://cdn.com/foo/bar/img.gif>

Content Delivery System

- DNS redirection
  - Vulnerable to position of DNS server

Content Delivery System

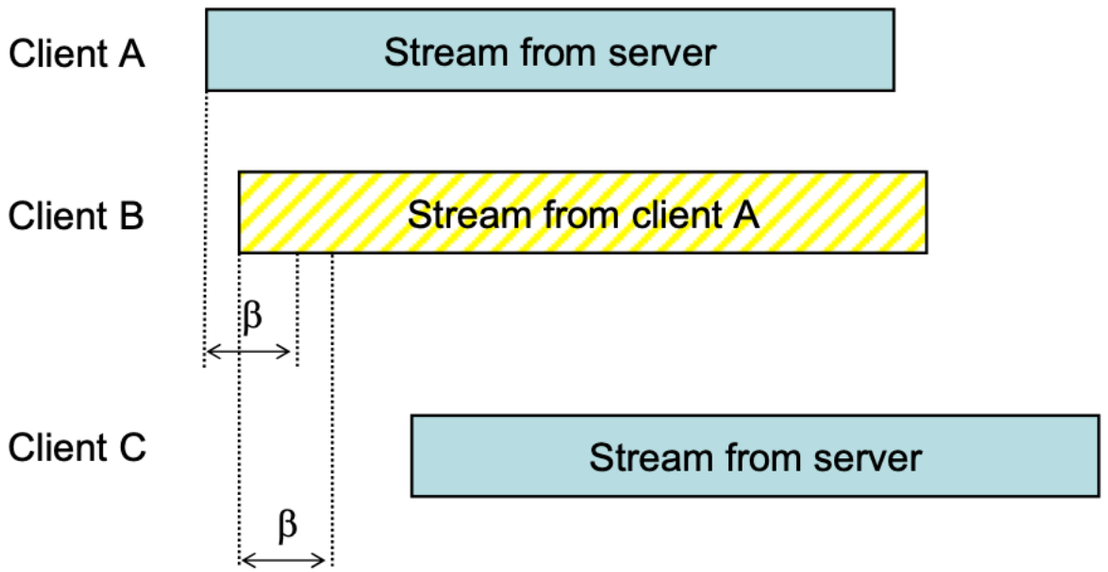
- URL rewrite

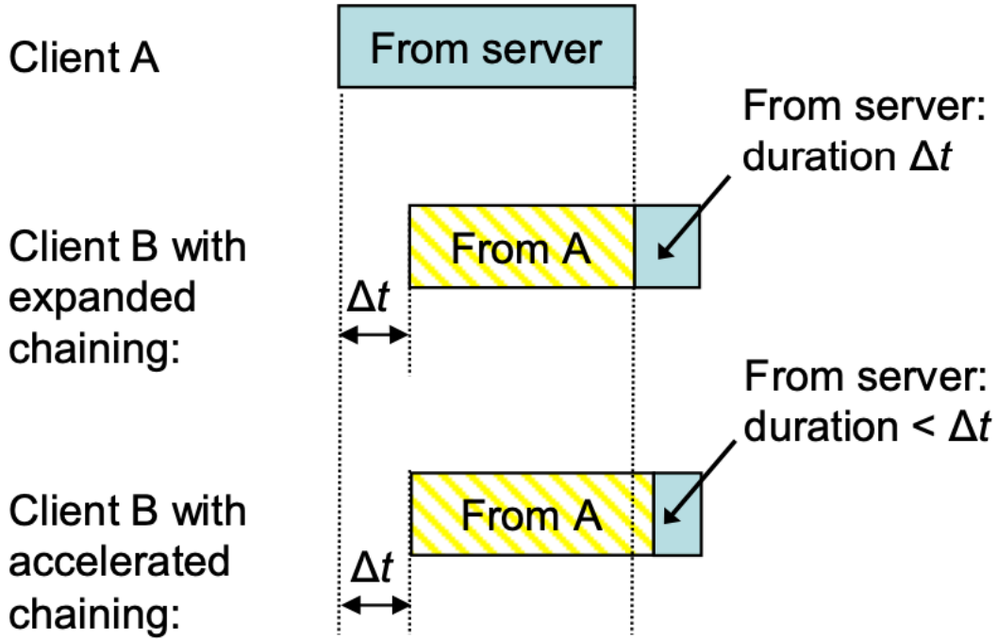
index.html

```
<HTML>
<BODY>
<A HREF="/about_us.html"> About Us </A>
<IMG SRC="www.clearway1.net/www.yahoo.com/img1.gif">
<IMG SRC="www.clearway2.net/www.yahoo.com/img2.gif">
<IMG SRC="10.20.30.2/www.yahoo.com/img3.gif">
</BODY>
</HTML>
```

Video on Demand

- Chaining:
  - Clients transfer data as well





Bit Torrent

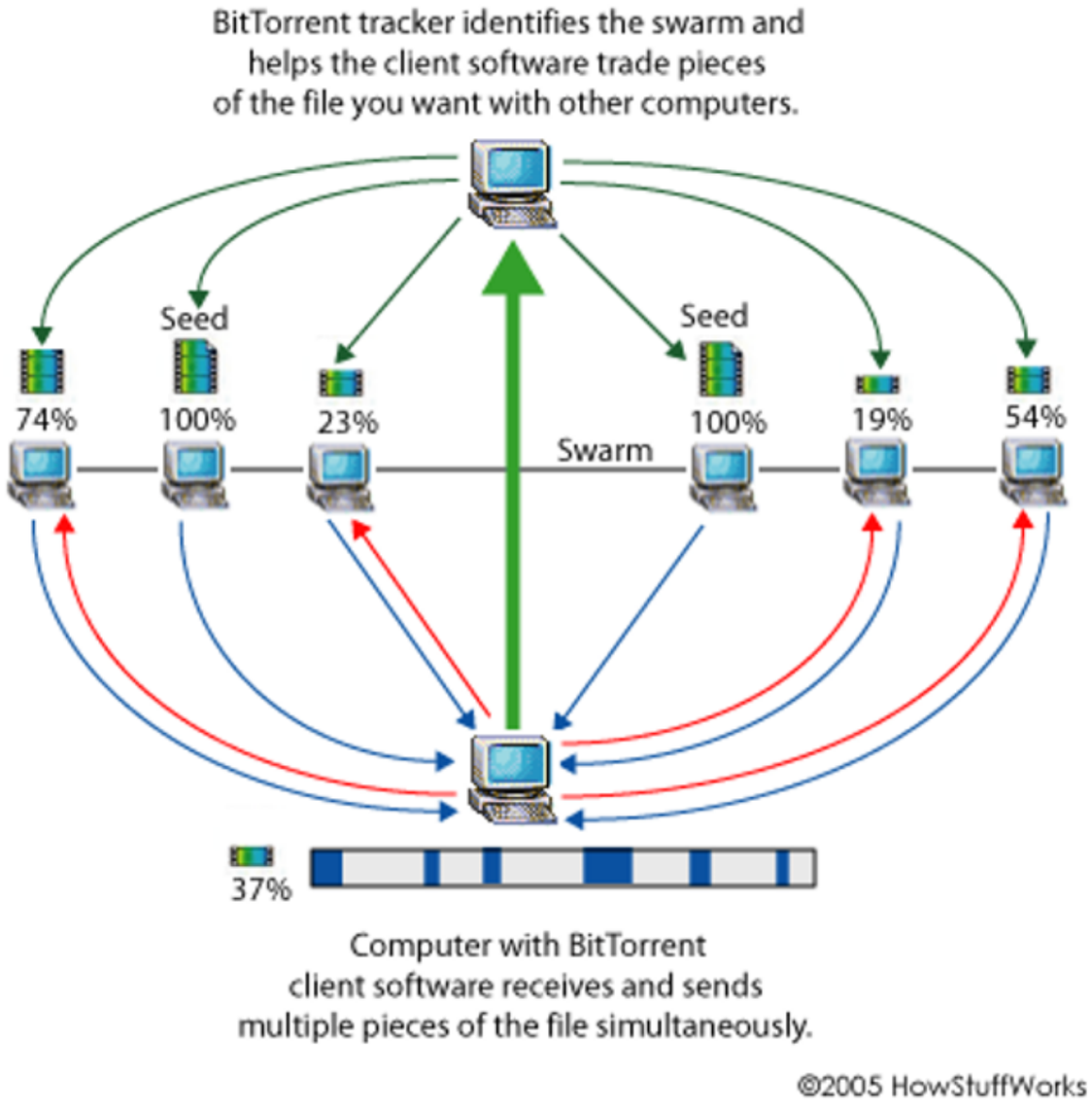
- Used for
  - Content distribution
  - Game Patching Systems
  - ...

Bit Torrent

- P2P System:
  - Peers:
    - Download files
    - Distribute to other peers
  - Seeds
    - Host files
    - Distribute to peers
    - Determine the speed of the download

Bit Torrent

- Trackers:
  - Delegate transfer speeds
  - Keep tabs on transfers
  - Block banned users
  - Block peers from abuse
  - Verify content availability



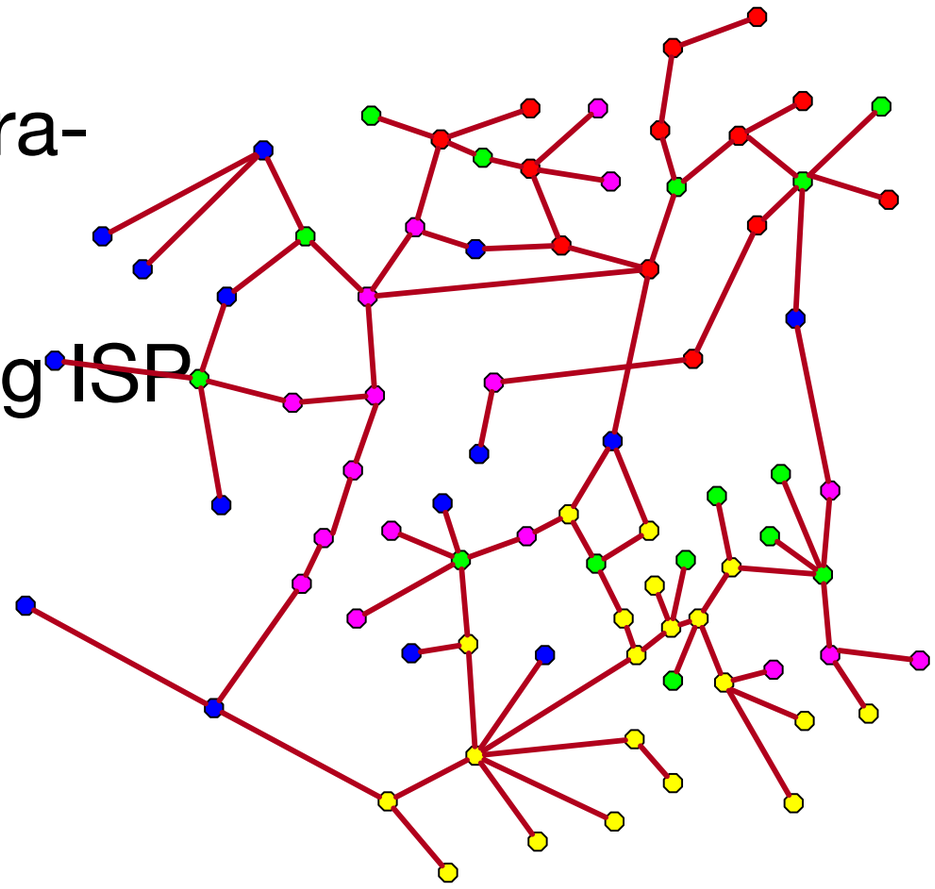
Traffic Shaping

Traffic Shaping Goals

- ISP limit peak traffic
  - Economic reasons:
    - Get charged for peak usage by other ISP for cross ISP traffic
    - Do not want to support rival technologies
      - E.g. Telephone company vs. other VoIP
- Quality of service:
  - Bulk, P2P receive lower priority

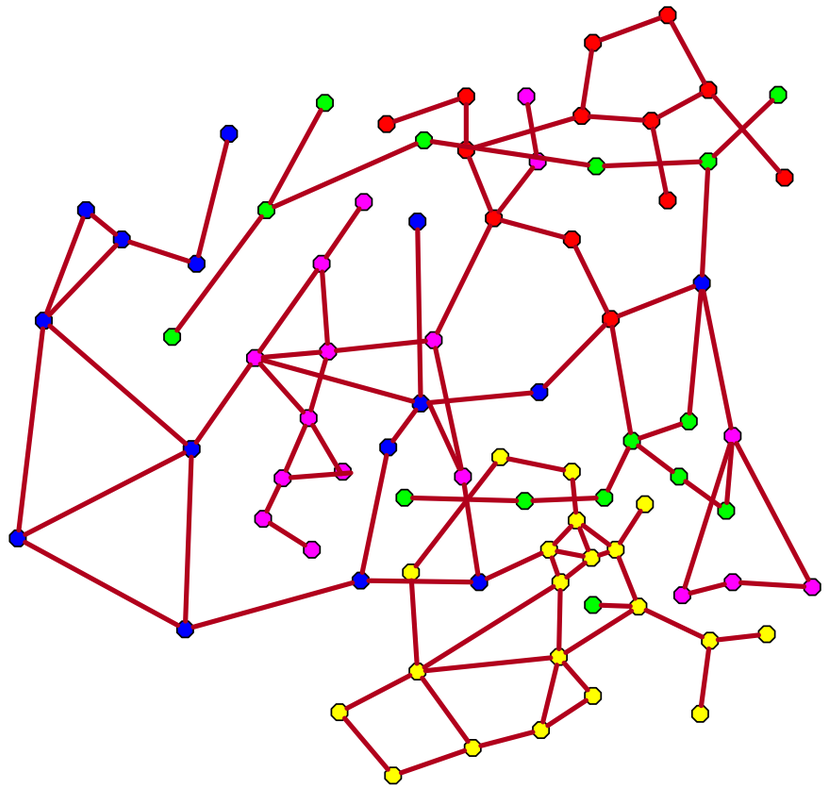
Controlling P2P Traffic

- P2P technology uses a virtual overlay network
  - Formed without regard to real infra-structure
  - Tends to have many links crossing ISP boundaries
    - And therefore incur charges



Controlling P2P Traffic

- Delay packages that set up the overlay network
  - If they cross ISP boundaries
  - P2P now forms an overlay network where most links are with the same ISP
- With some P2P:
  - Possible that overlay network components exist only inside one ISP network
- Result: Less charges for cross-ISP traffic



Controlling P2P Traffic

- Example: Gnutella
  - Gnutella user wants a file
    - Gnutella floods peers with queries
    - If peer finds it, starts sending file
  - ISP:
    - Delay query message to cross-ISP peers
      - Favor "own" peers: More likely to be selected for download
    - Result:
      - Bulk traffic within ISP
      - Gnutella might break into sub-Gnutellas, each within an ISP

Controlling P2P Traffic

- BitTorrent:
  - Users that download the same resource form a swarm
    - Each user who downloads also uploads to later members of the swarm
  - Used by Blizzard to distribute its games
  - But piracy creates legal problems for ISP
  - By slowing down BitTorrent traffic:
    - ISP can discourage its use

Controlling Streaming

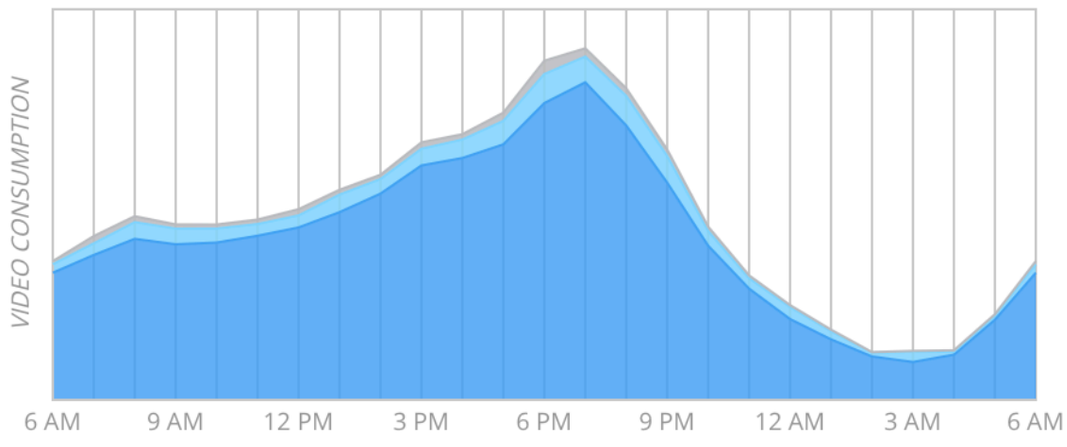
AT&T - Other in Washington County Change Location

Standard Definition

Users on networks rated as Standard Definition should expect smooth playback on standard-definition YouTube videos (360p) and may experience occasional interruptions on high-definition YouTube videos (720p and above).

If you're experiencing issues playing your video, try these troubleshooting tips.

VIDEO CONSUMPTION AND STREAMING QUALITY



- Lower Definition (LD) streams
- Standard Definition (SD) streams
- High Definition (HD) streams

Daily video activity is averaged over 30 days.

Individual results may vary and the results for the same ISP may also vary between different locations and service speeds (methodology and FAQ).

Throttling

- Routers can use deep packet inspection
  - Determines type of traffic
  - Set up rules:
    - Time of day
    - Network load
    - User behavior
- Use rules based on result
  - blocking or inserting a TCP Fin / TCP Res package
  - Drop packets
  - Modify TCP advertised window size -> sender slows down
  - Modify protocol messages

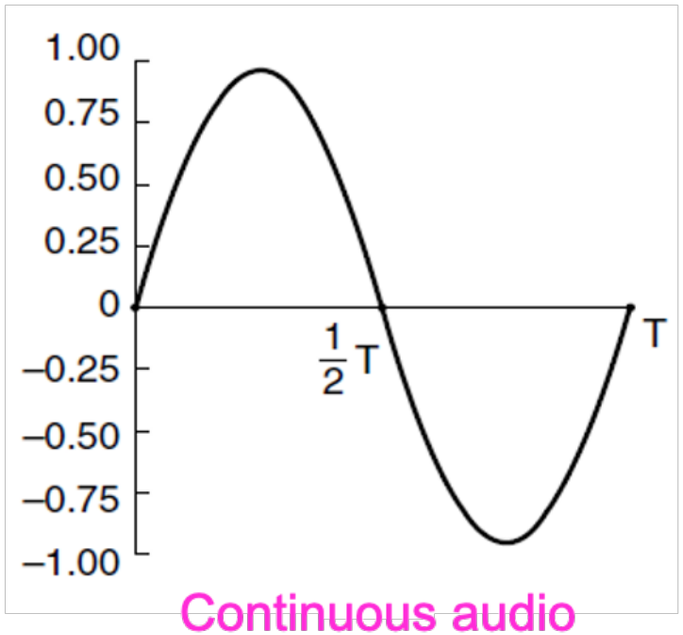
Streaming Audio and Video

Streaming Audio and Video

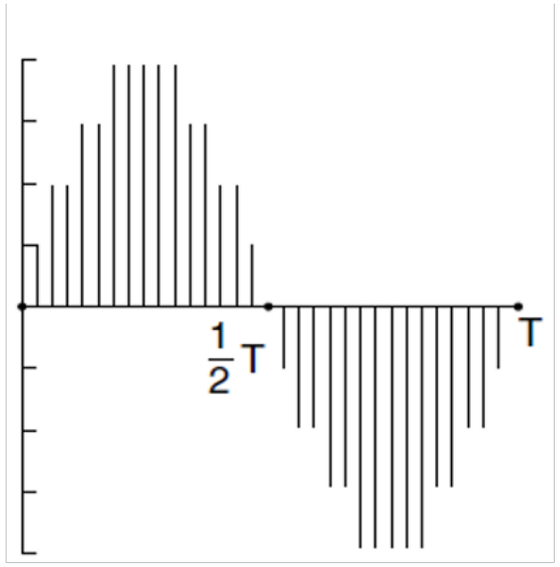
- Audio and video have become key types of traffic, e.g., voice over IP, and video streaming.
  - Digital audio
  - Digital video
  - Streaming stored media
  - Streaming live media
  - Real-time conferencing

Digital Audio

- ADC (Analog-to-Digital Converter) produces digital audio from a microphone
  - Telephone: 8000 8-bit samples/second (64 Kbps)
  - Computer audio is usually better quality (e.g., 16 bit)



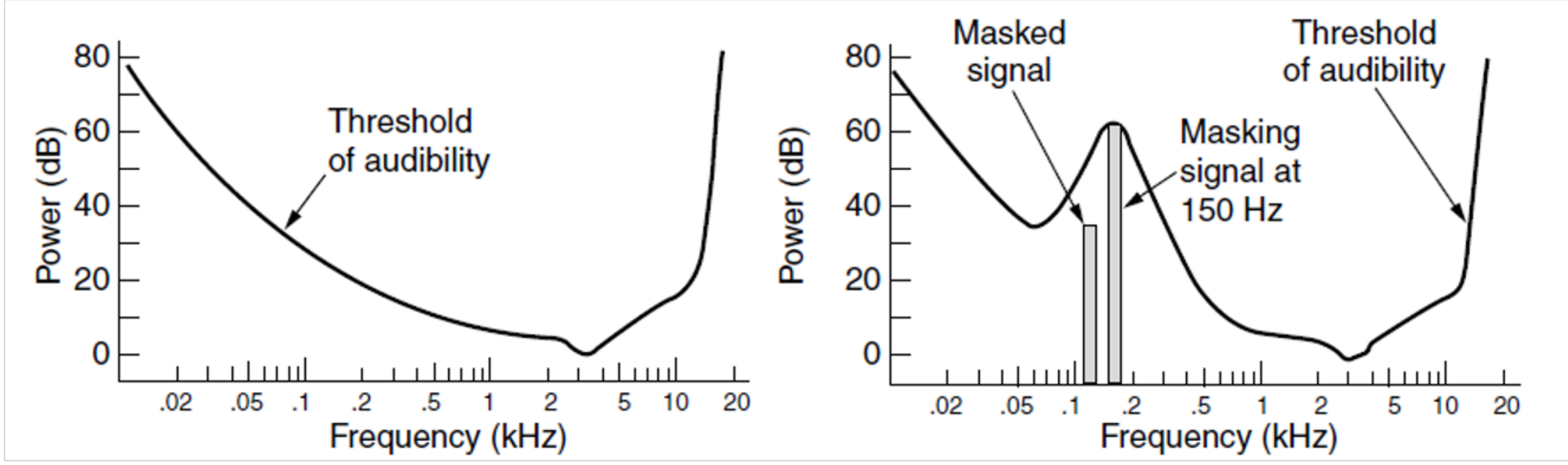
Continuous audio
(sine wave)



Digital audio
(sampled, 4-bit quantized)

Digital Audio

- Digital audio is typically compressed before it is sent
  - Lossy encoders (like AAC) exploit human perception
  - Yields large compression ratios (can be >10X)



Sensitivity of the ear varies with frequency

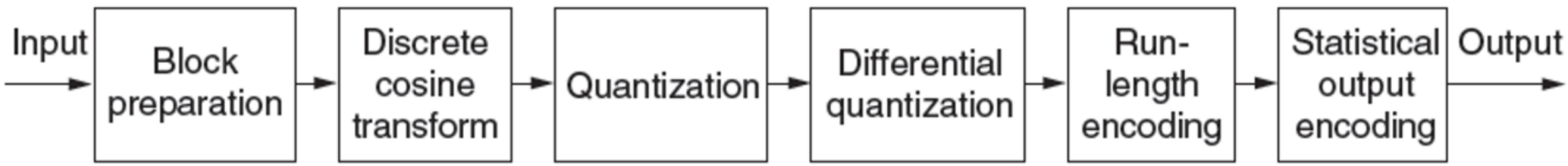
A loud tone can mask nearby tones

Digital Video

- Video is digitized as pixels (sampled, quantized)
  - TV quality: 640x480 pixels, 24-bit color, 30 times/sec
  - Video is sent compressed due to its large bandwidth
    - Lossy compression exploits human perception
      - E.g., JPEG for still images, MPEG, H.264 for video
      - Large compression ratios (often 50X for video)
- Video is normally > 1 Mbps, versus >10 kbps for speech and >100 kbps for music

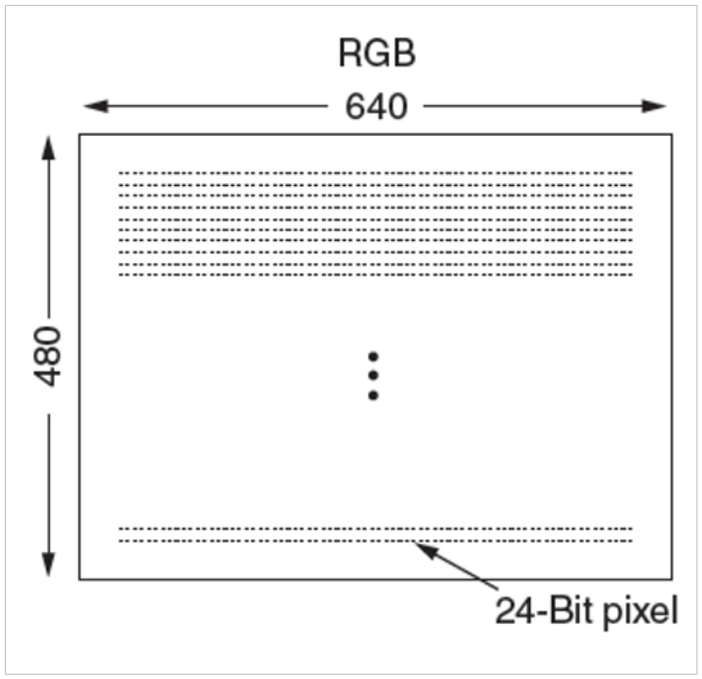
Digital Video

- JPEG Image Compression

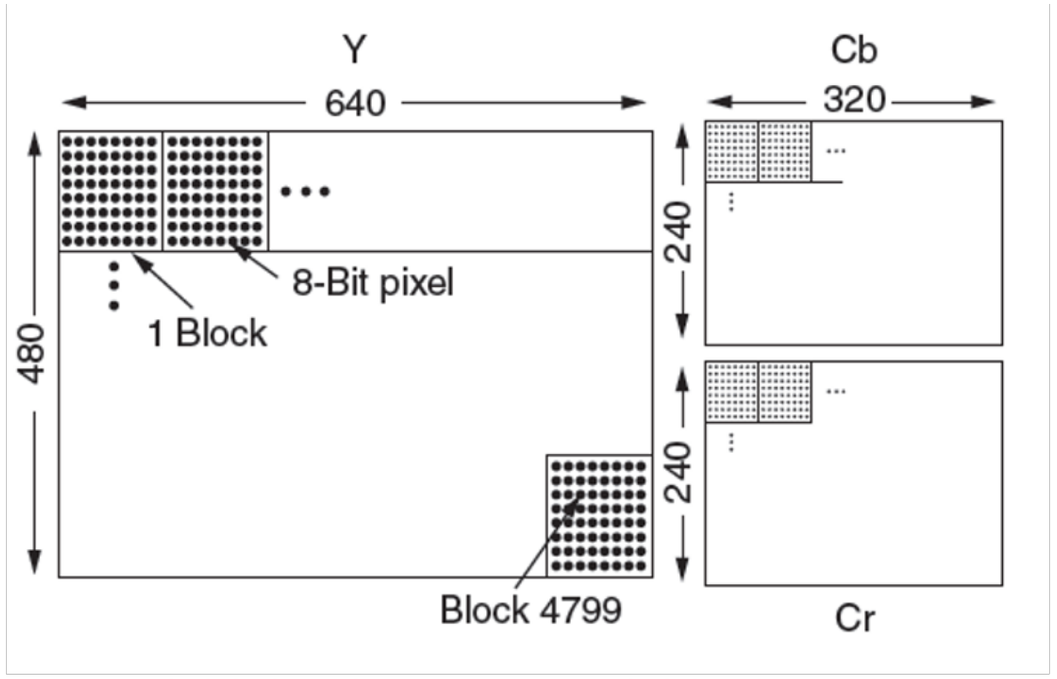


Digital Video

- Step 1: Map pixels to luminance /chrominance color space
  - *Sub-sample* chrominance



Input 24-bit RGB pixels

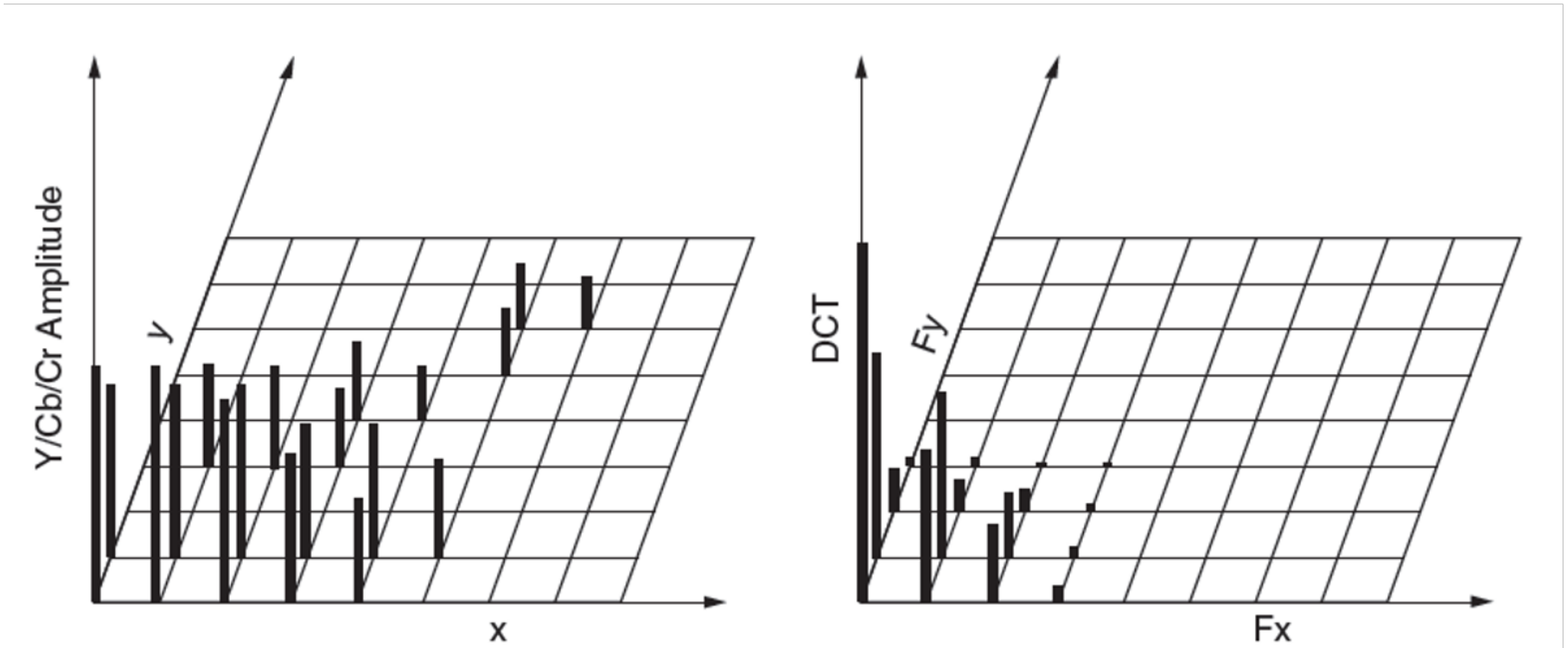


8-bit luminance pixels

8-bit chrominances for every 4 pixels

Digital Video

- Step 2: Transform each block with Discrete Cosine Transformation (DCT)



One component block

Transformed block

Digital Video

- Step 3: Divide DCT components by thresholds
- Step 4: Use a “quantization table” to reduce the number of bits in the DCT representation

DCT coefficients

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Input

/

Quantization table

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

Thresholds

=

Quantized coefficients

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Output

Digital Video

- Example:
  - Reduction to 5%

Original Compressed

Digital Video

- Step 5: Use a zig-zag pattern to replace the 2d array with a 1d array
- Step 6: Use a Huffman encoding on the values

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Order in which the block coefficients are sent

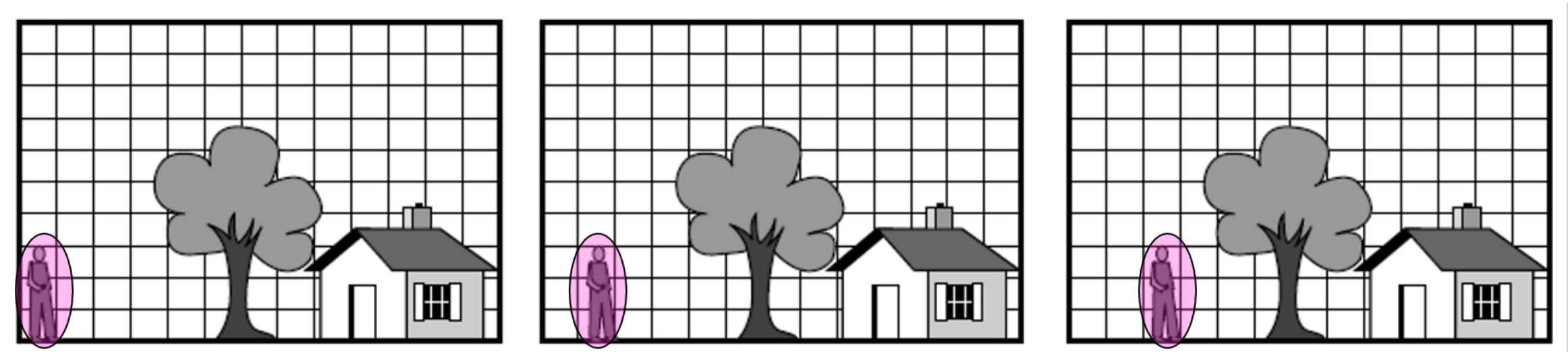
Digital Video

- Huffman codes:
  - Various length code
  - Frequent patterns are encoded with shorter bit strings

Digital Video

- MPEG
  - Variety of different compression standards
  - General Technique:
    - Three types of frames:
      - I (Intra-coded) frames are self-contained
      - P (Predictive) frames use block motion predictions
      - B (Bidirectional) frames may base prediction on future frame

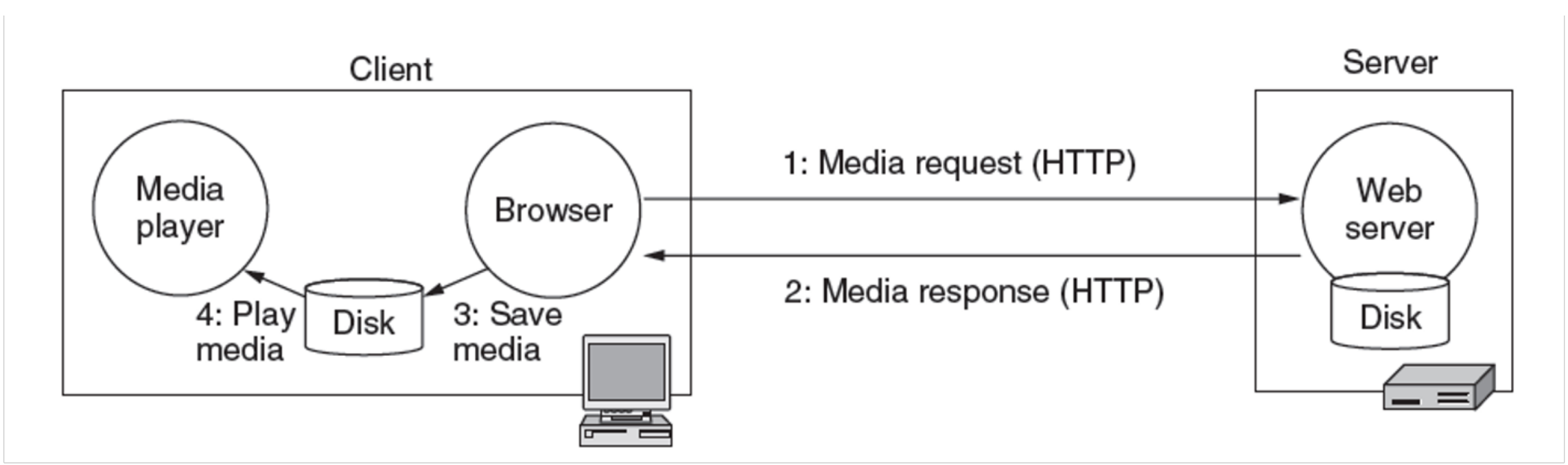
Digital Video



Three consecutive frames with stationary and moving components

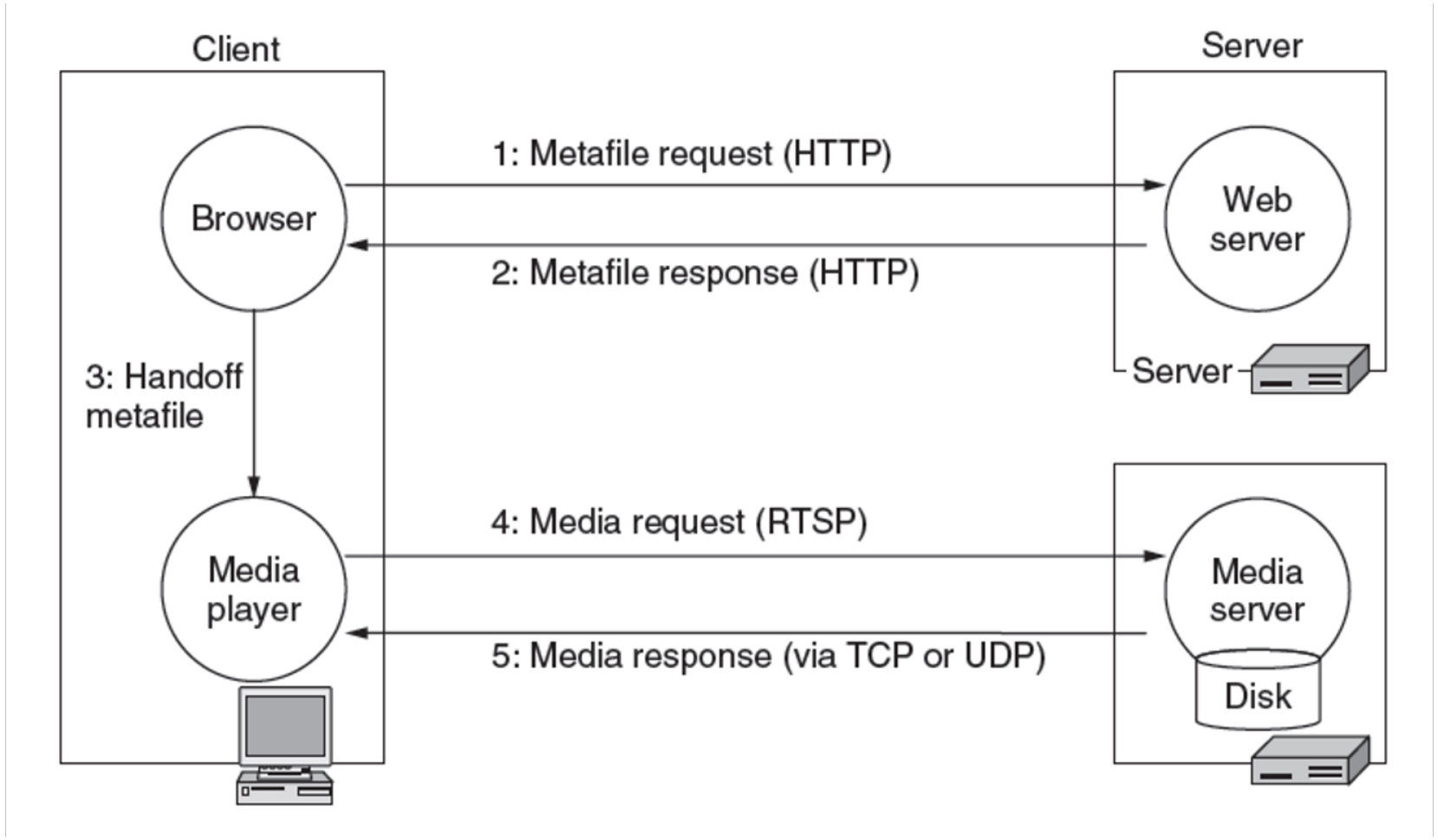
Streaming Stored Content

- A simple method to stream stored media, e.g., for video on demand, is to fetch the video as a file download
  - But has large startup delay, except for short files



Streaming Stored Content

- Effective streaming starts the playout during transport
  - With RTSP (Real-Time Streaming Protocol)



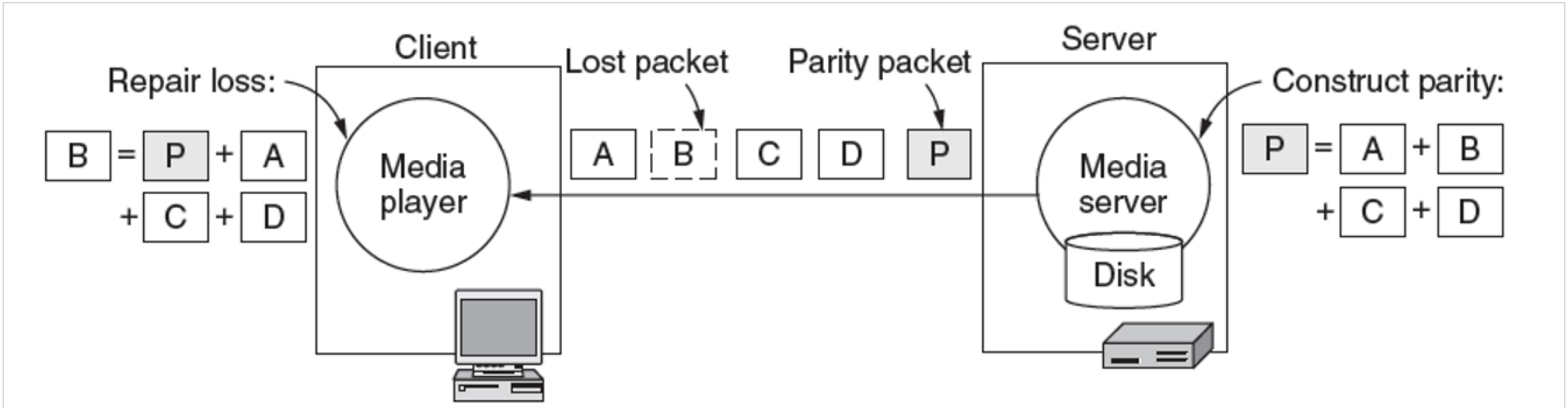
Streaming Stored Content

- Key Problem: How to deal with transmission errors

Strategy	Advantage	Disadvantage
Use reliable transport (TCP)	Repairs all errors	Increases jitter significantly
Add Forward Error Correction (e.g., parity)	Repairs most errors	Increases overhead, decoding complexity and jitter
Interleave media	Masks most errors	Slightly increases decoding complexity and jitter

Streaming Stored Content

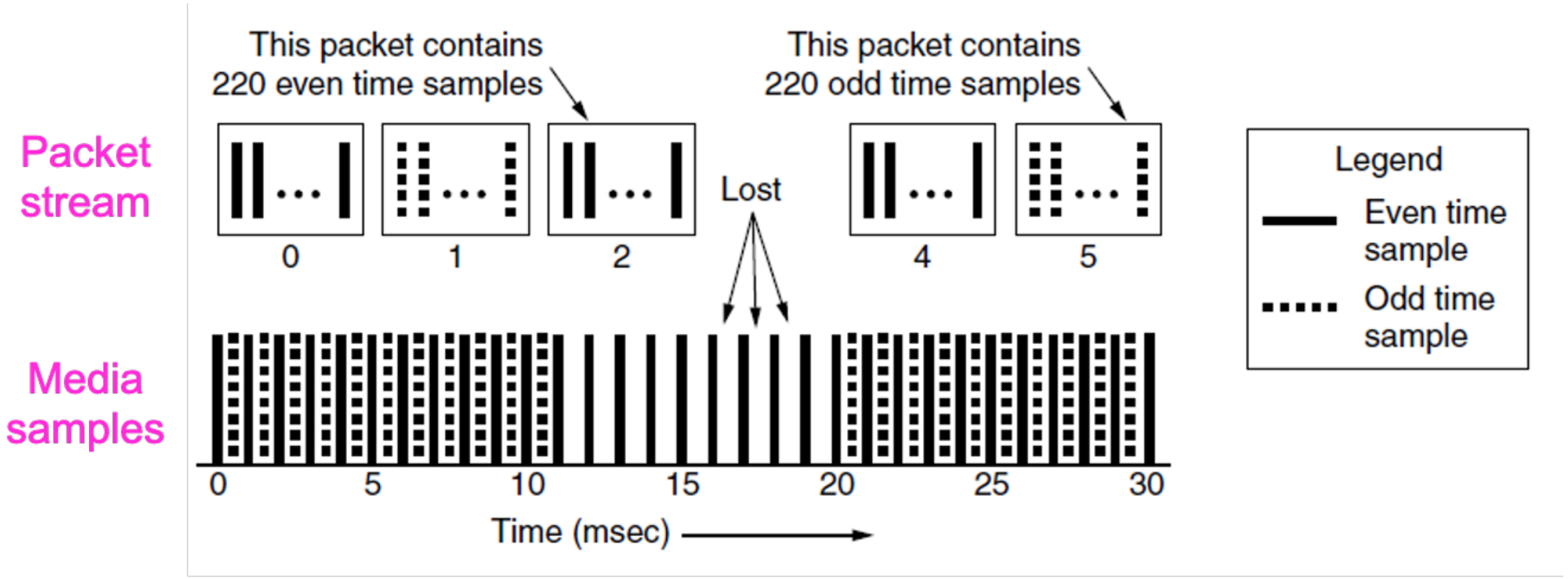
- Example: Include parity frame



- Need to buffer longer because all frames in the reliability group are needed

Streaming Stored Content

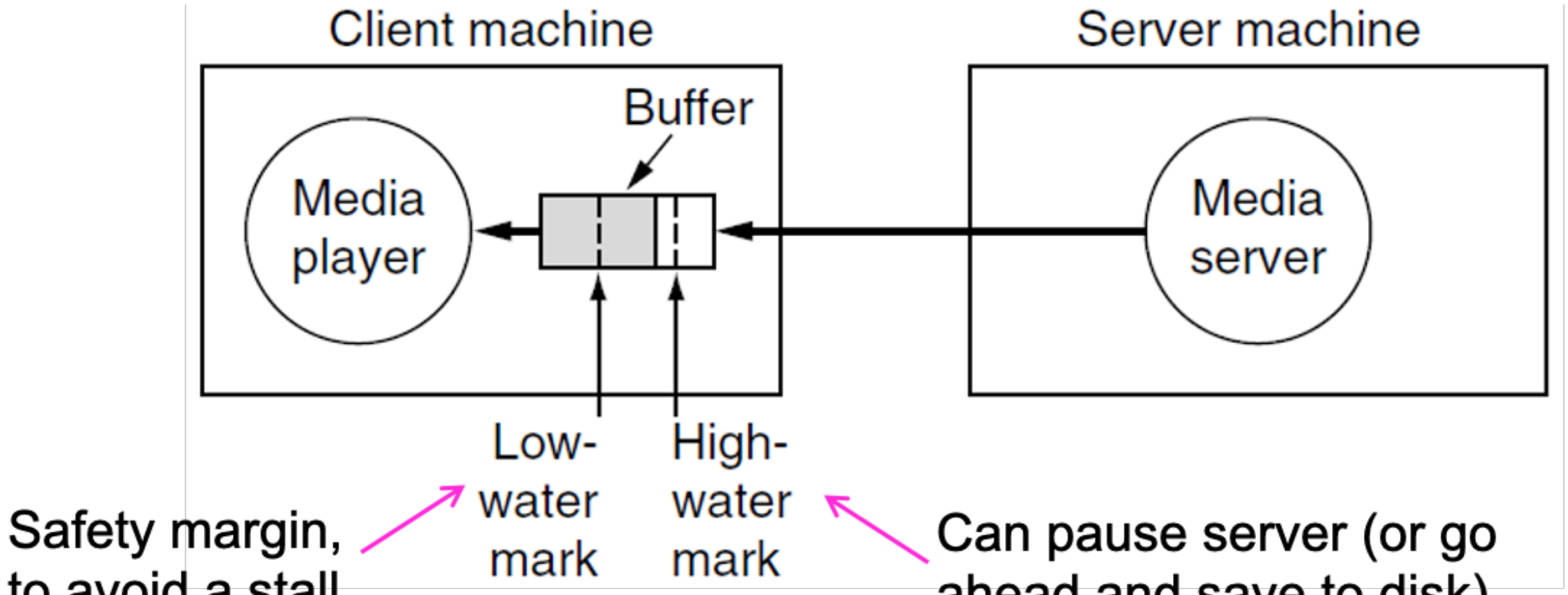
- Can use interleaving
  - Segments belong to an even and an odd stream
  - Reduces impact of losses



Loss reduces temporal resolution; doesn't leave a gap

Streaming Stored Content

- To prevent jitter: Need to buffer
  - Match service rate to display rate over time



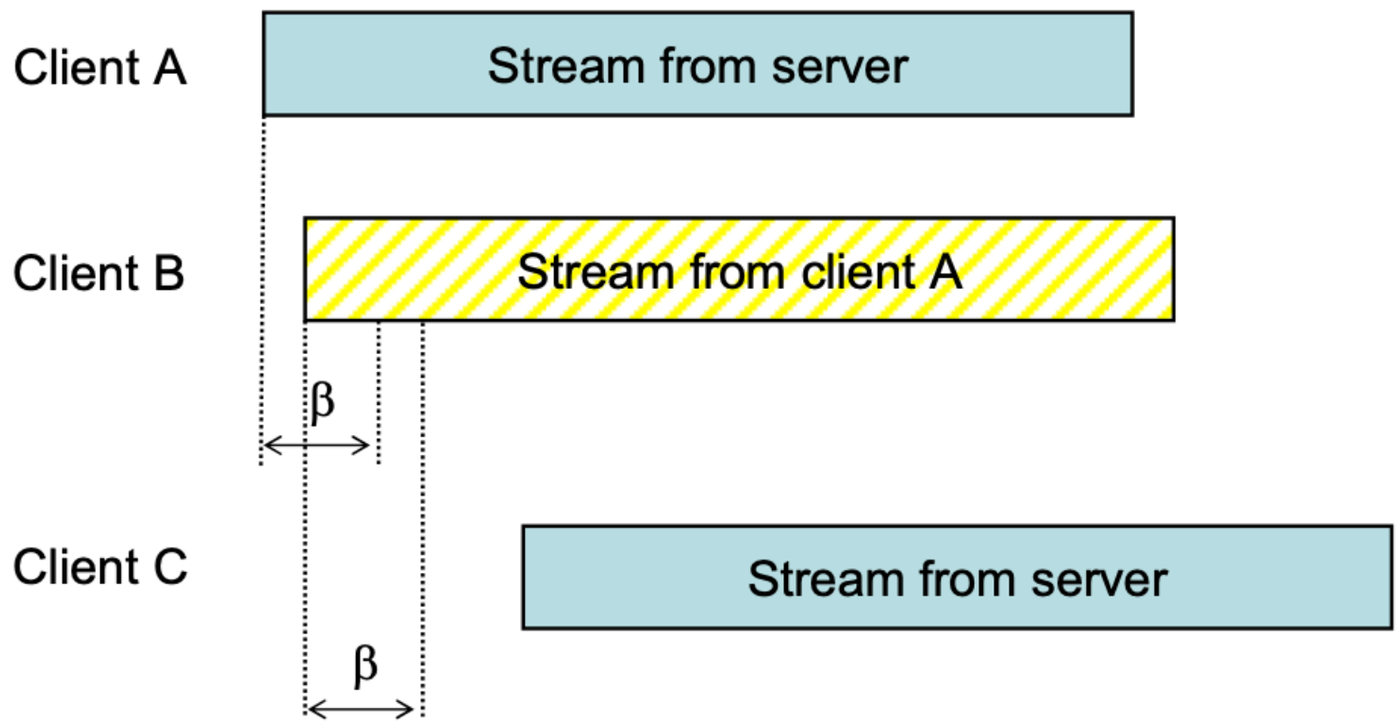
Streaming Stored Content

- RTSP commands
  - Sent from player to server to adjust streaming

Command	Server action
DESCRIBE	List media parameters
SETUP	Establish a logical channel between the player and the server
PLAY	Start sending data to the client
RECORD	Start accepting data from the client
PAUSE	Temporarily stop sending data
TEARDOWN	Release the logical channel

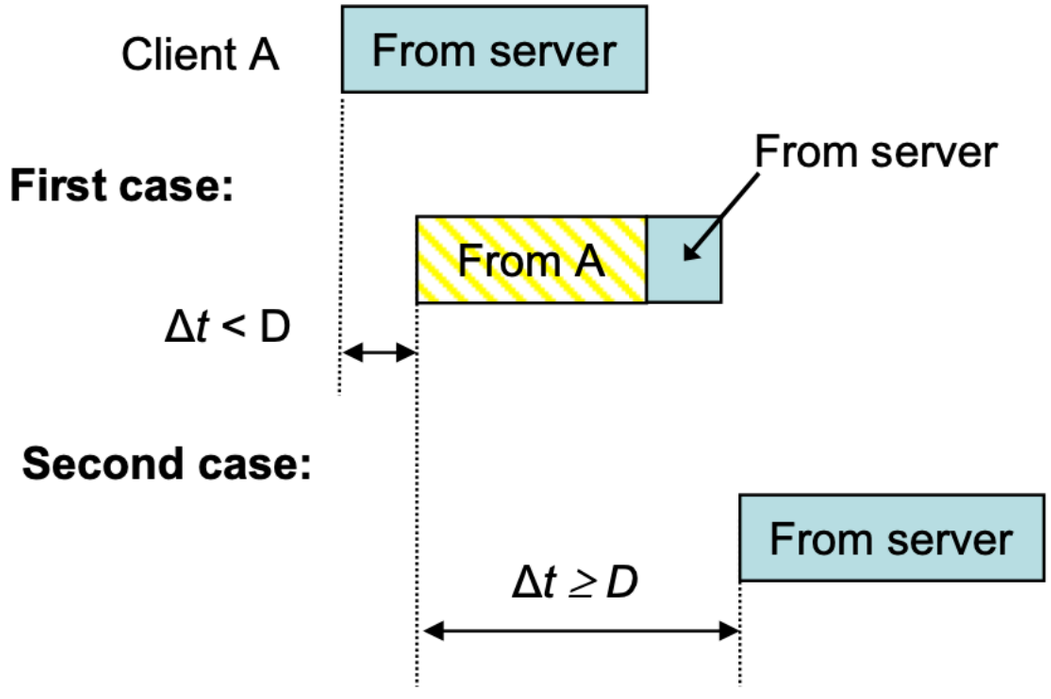
Streaming Stored Content

- P2P solutions:
  - Reduce server farm by forcing clients to be servers as well
  - Chaining protocols



Streaming Stored Content

- Expanded chaining allows the client to stop serving when they no longer consume



Streaming Stored Content

- Bit-Torrent
  - Clients (peers) forward all content to other peers
  - Files are broken into small pieces (chunks)
  - Downloaded to users in a group (swarm)
  - While downloading, users upload to others in the swarm

Streaming Stored Content

- Bit-Torrent
  - Organize data transfer: which peers have what, who do we serve chunks
  - Churn: Peers come and go
  - Fairness: Leechers do not upload

Streaming Stored Content

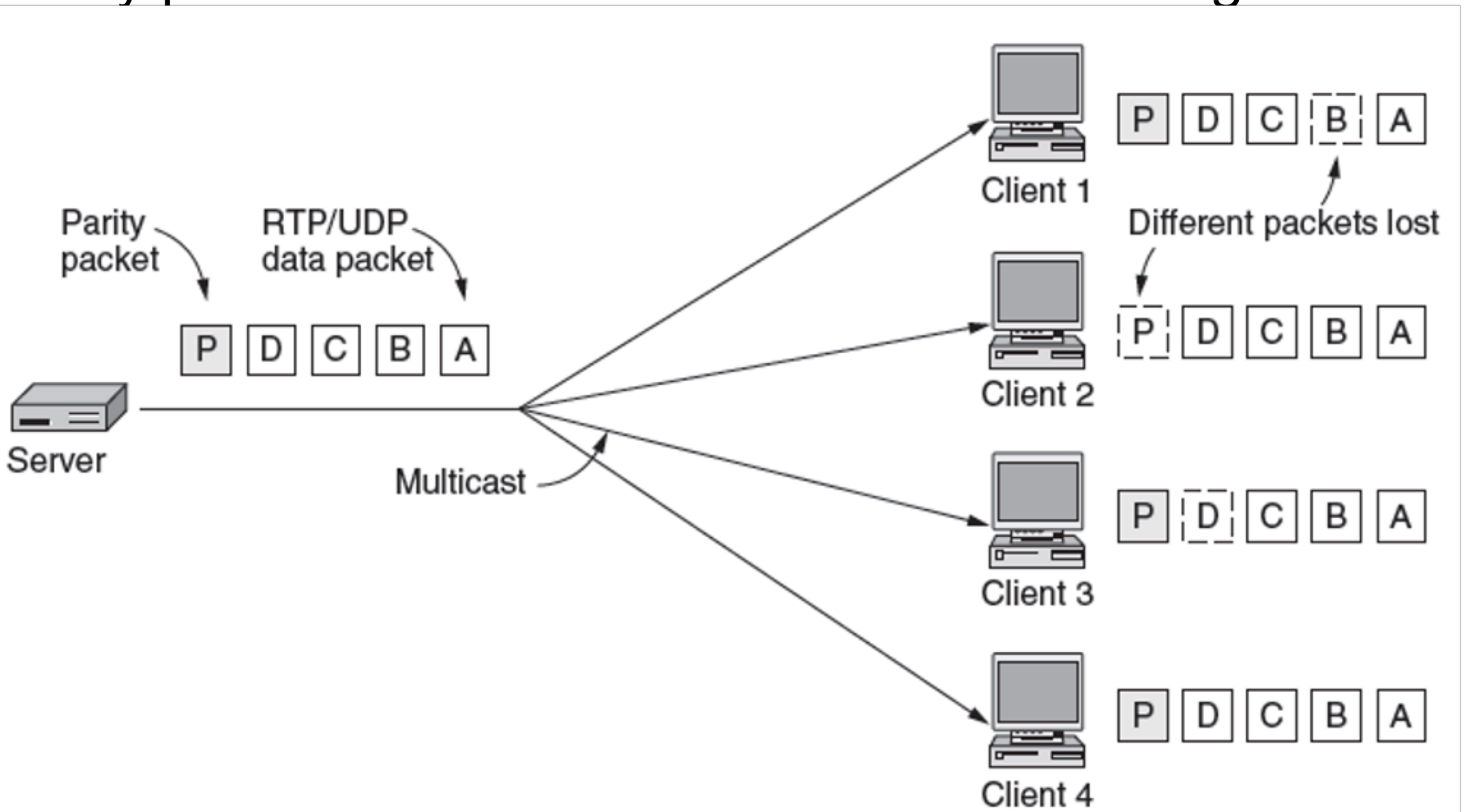
- Bit-Torrent
  - Centralized tracker
    - Collects information on peers
    - Responds to requests
  - Rechoking favors cooperative peers

Streaming Live Content

- Streaming live media is similar to the stored case plus:
  - Can’t stream faster than “live rate” to get ahead
  - Usually need larger buffer to absorb jitter
  - Often have many users viewing at the same time
    - UDP with multicast greatly improves efficiency. It is rarely available, so many TCP connections are used.
    - For very many users, content distribution is used [later]
-

Streaming Live Content

- Parity protection more efficient with multicasting



Streaming Live Content