Network Security

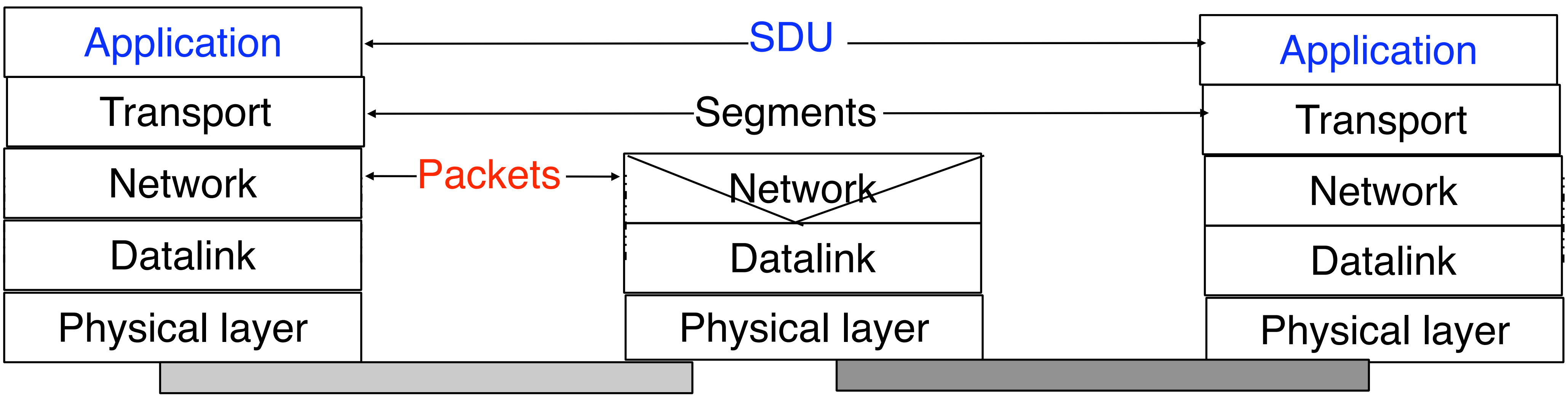
Olivier Bonaventure

IP Networking Lab
Department of Computing Science and Engineering
Université catholique de Louvain (UCL)
Place Sainte-Barbe, 2, B-1348, Louvain-la-Neuve (Belgium)

http://inl.info.ucl.ac.be

Part 2 : Networking

Network security

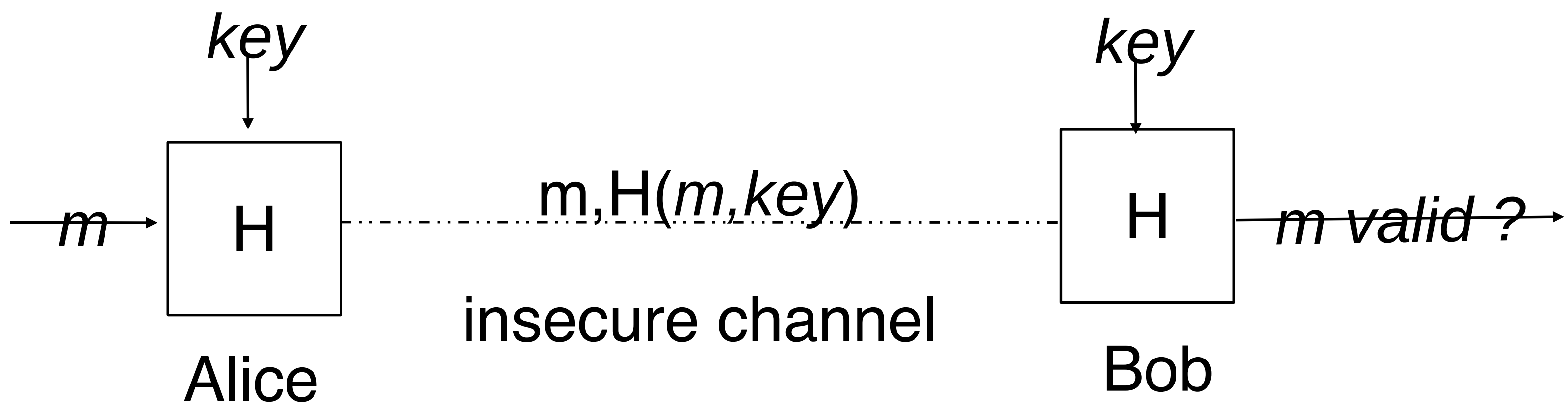


- At which layer should we place the security functions ?

- Crypto building blocks
- Application-layer security
  - Secure Socket Layer
- Transport-layer security
- Network-layer security

Cryptographical building blocks
Hash functions

- Hash functions

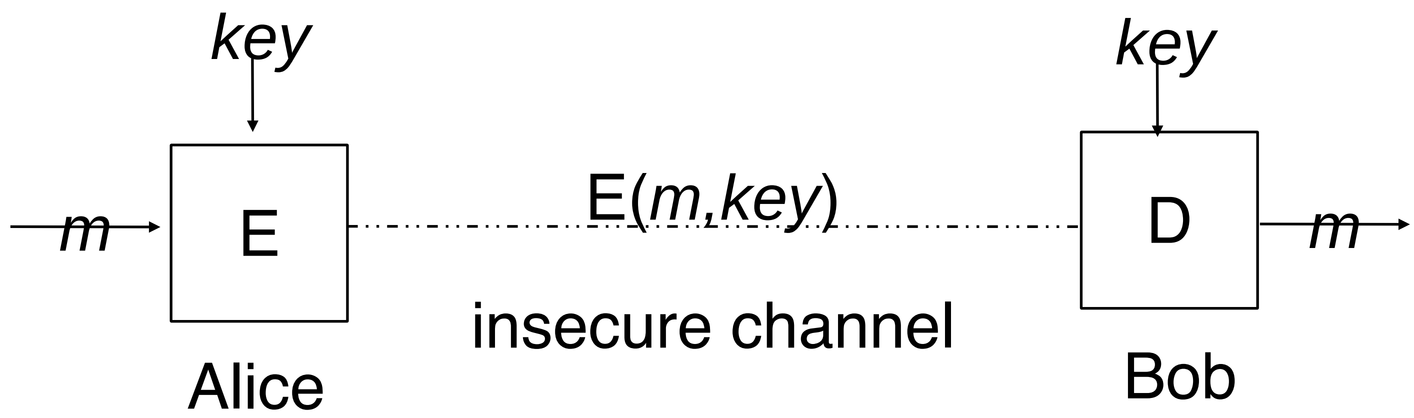


  - Properties
    - Easy to compute H(Msg,key)
    - Very difficult to find Msg2 : H(Msg,k)=H(Msg2,k)

- Example hash functions
  - MD5, MD4, SHA-1

Cryptographical building blocks
Secret-key cryptography

- Secret-key cryptography



- Advantages
  - Efficient algorithms exist
  - Security is f(implementation and key size)
- Drawbacks
  - Key must be distributed securely
  - Does not provide any authentication scheme
- Examples : DES, AES, RC-4, IDEA,...

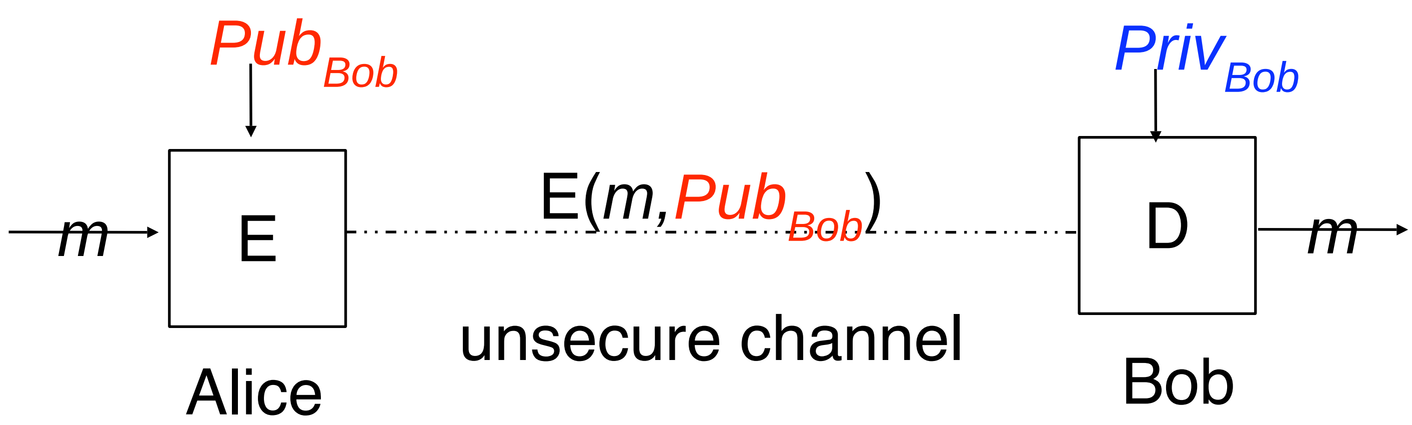
A detailed description of (too) many cryptographical algorithms may be found in :
B. Scheneier, Applied Cryptography, second edition, Wiley, 1995

A more concise description appears in :

C. Kaufman, R. Perlman and M. Speciner, Network Security : Private Communications in a public world, Prentice Hall, second edition, 2002

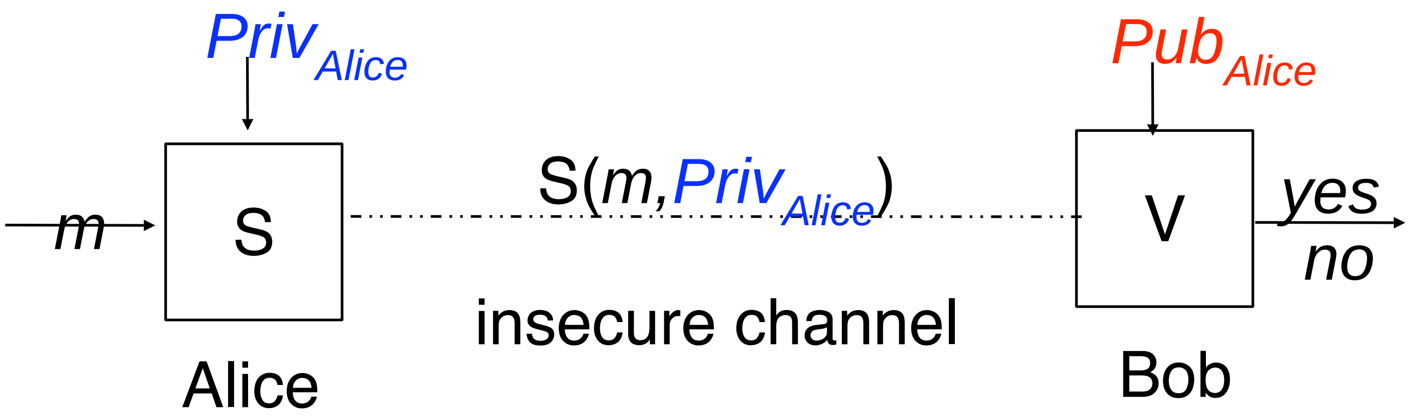
Cryptographical building blocks
Public-key cryptography

- Public-key cryptography
  - Each user maintains two keys
    - A public key ($Public_{Key}$) which can be made public and can be used by any user to send him/her encrypted messages
    - A private key ($Private_{Key}$) which is kept secret and can be used to decrypt information encrypted with the public



Cryptographical building blocks
Public-key cryptography (2)

- Advantages
  - Users do not need to share a secret key to be able to encrypt messages
  - Public-key cryptography allows signatures



  - Security is f(implementation and key size)
- Drawbacks
  - Public-key cryptography is 10 or 100 times slower than secret-key cryptography
- Examples : RSA, DSS

Internet and Network security

- Crypto building blocks
- Application-layer security
  - Secure Socket Layer
- Transport-layer security
- Network-layer security

Building a simple secure protocol suitable for e-Commerce applications

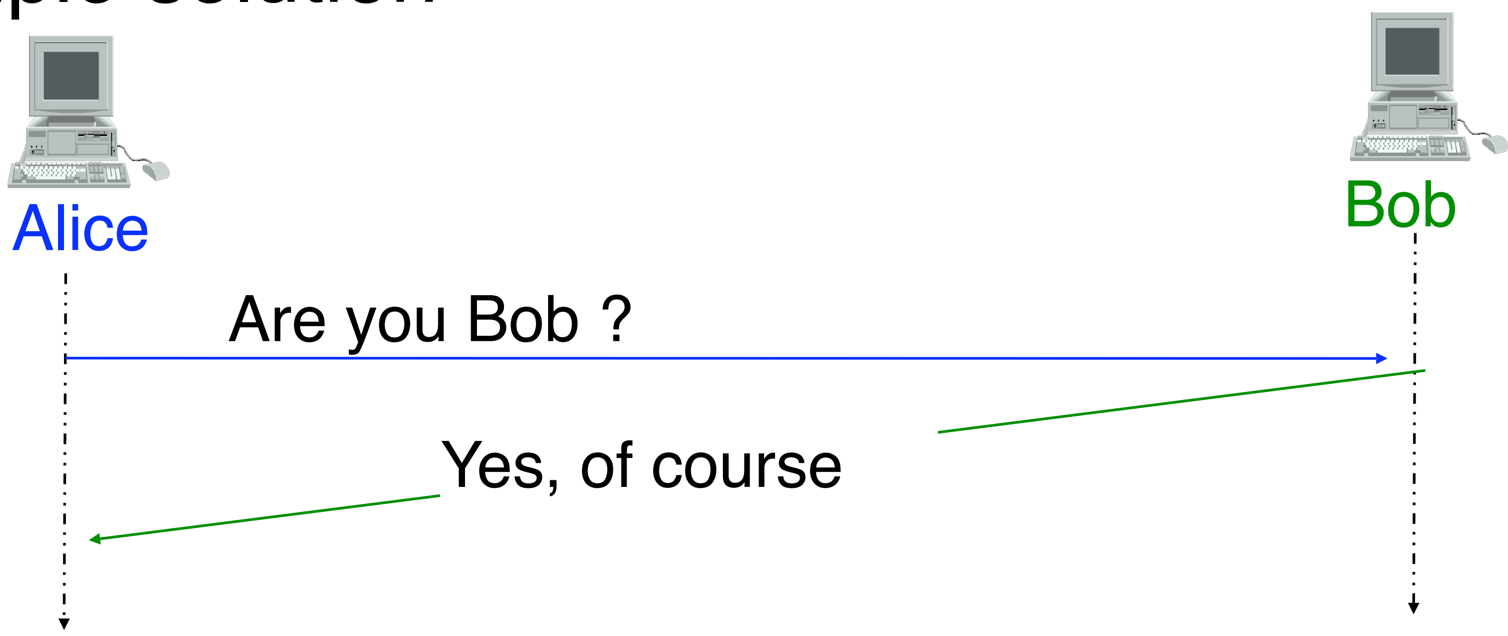
- Problems to solve
  - How to authenticate the server ?
  - How to authenticate the client ?
  - How to agree on an encryption key ?
  - How to encrypt data ?

This section is partially based on :

C. Kaufman, R. Perlman, M. Speciner, Network Security : Private communication in a Public world, Prentice Hall

How to authenticate the server ?

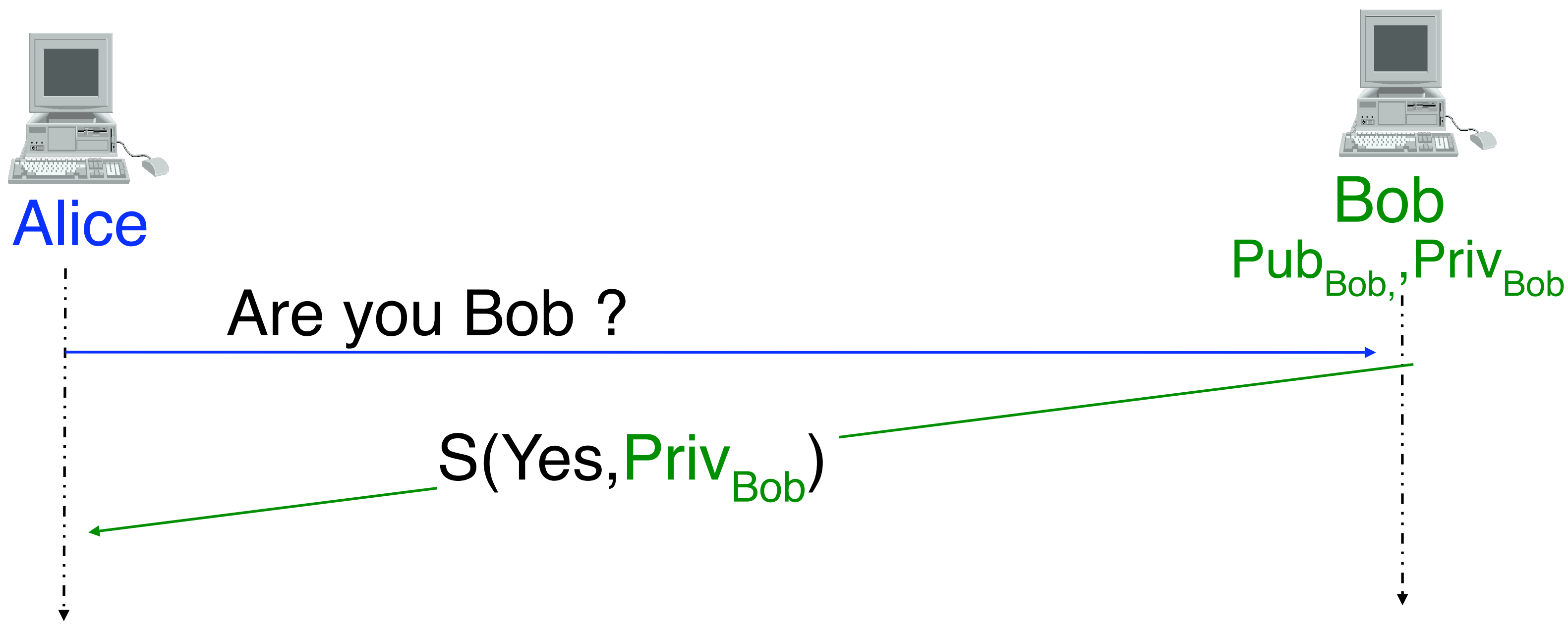
- Simple solution



- A more secure protocol is necessary
- Principle
  - Each server maintains a (public,private) key pair
  - Pub_{Bob} , $Priv_{Bob}$

The public-private key pair can be a RSA key-pair for example.

How to authenticate the server (2) ?

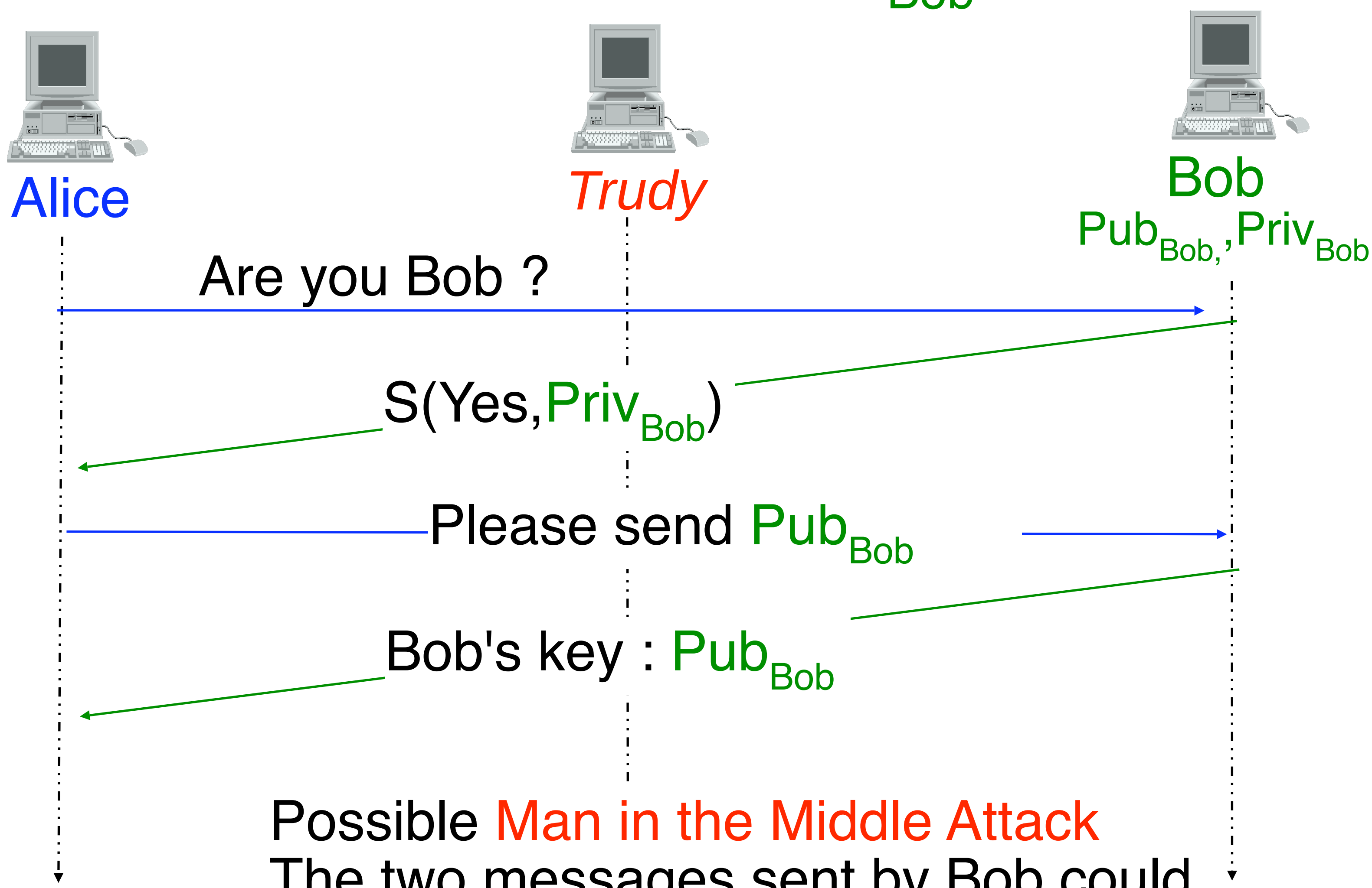


- Is this a secure authentication ?
  - Alice must already know Pub_{Bob}

In this slide and the subsequent ones,S(Yes,$Priv_{Bob}$) is a signed message that contains “Yes” and is signed by using the ,$Priv_{Bob}$ private key . The validity of this signature can be checked by using ,Pub_{Bob}

How to authenticate the server (3) ?

- Can Alice ask Bob for Pub_{Bob} ?

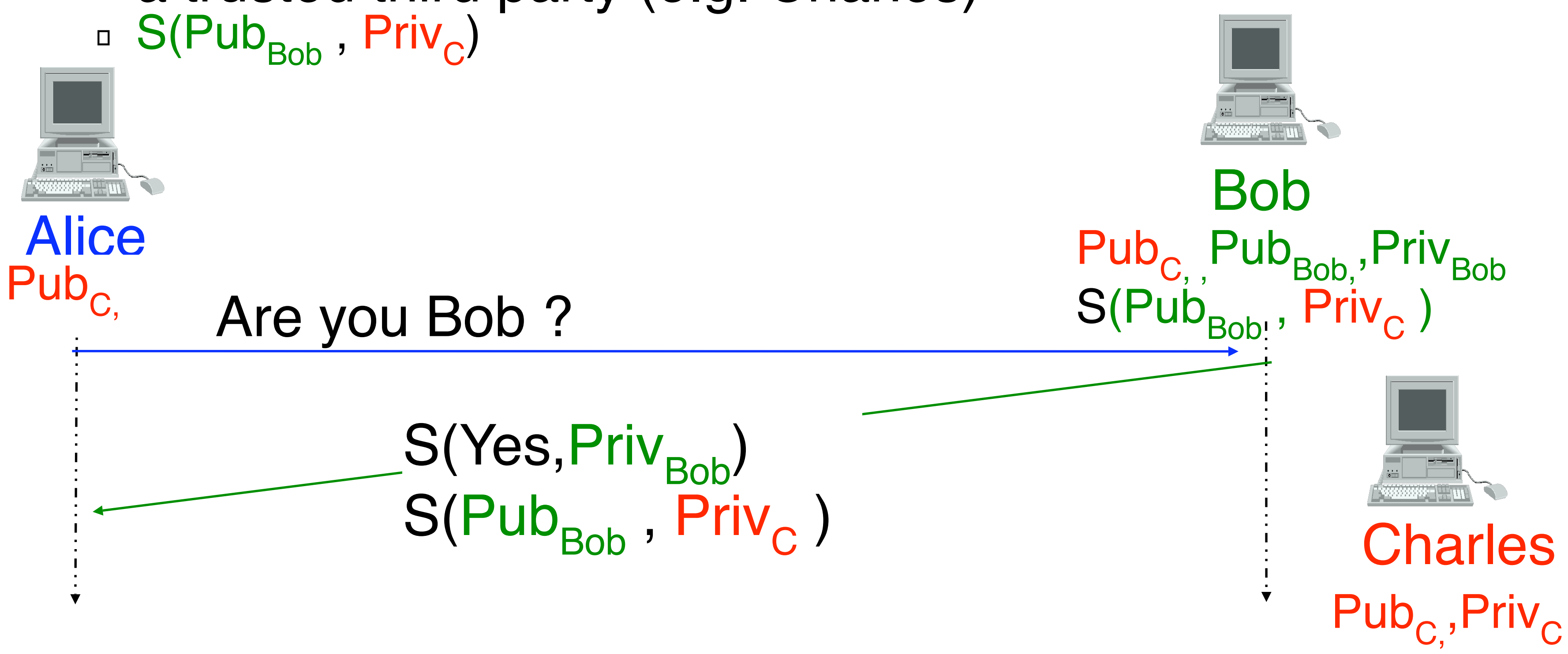


Possible Man in the Middle Attack
The two messages sent by Bob could also have been sent by Trudy

A Man in the Middle or Woman in the Middle attack is possible in this case as Trudy can easily intercept the messages sent by Alice and replace them with fake messages that contain her public key and signature.

How to authenticate the server (4) ?

- Public-key certificates
  - To authenticate public keys, Alice and Bob must trust a third party
  - Certificate
    - information about a user/server and public-key signed by a trusted third party (e.g. Charles)
      - $S(Pub_{Bob}, Priv_C)$



In the example above, we use $S(Pub_{Bob}, Priv_C)$ to indicate a certificate for Bob's key issued by Charles.

Charles usually checks the identity of Bob offline and then creates the certificate. Charles is sometimes referred to as a Trusted Third Party (TTP).

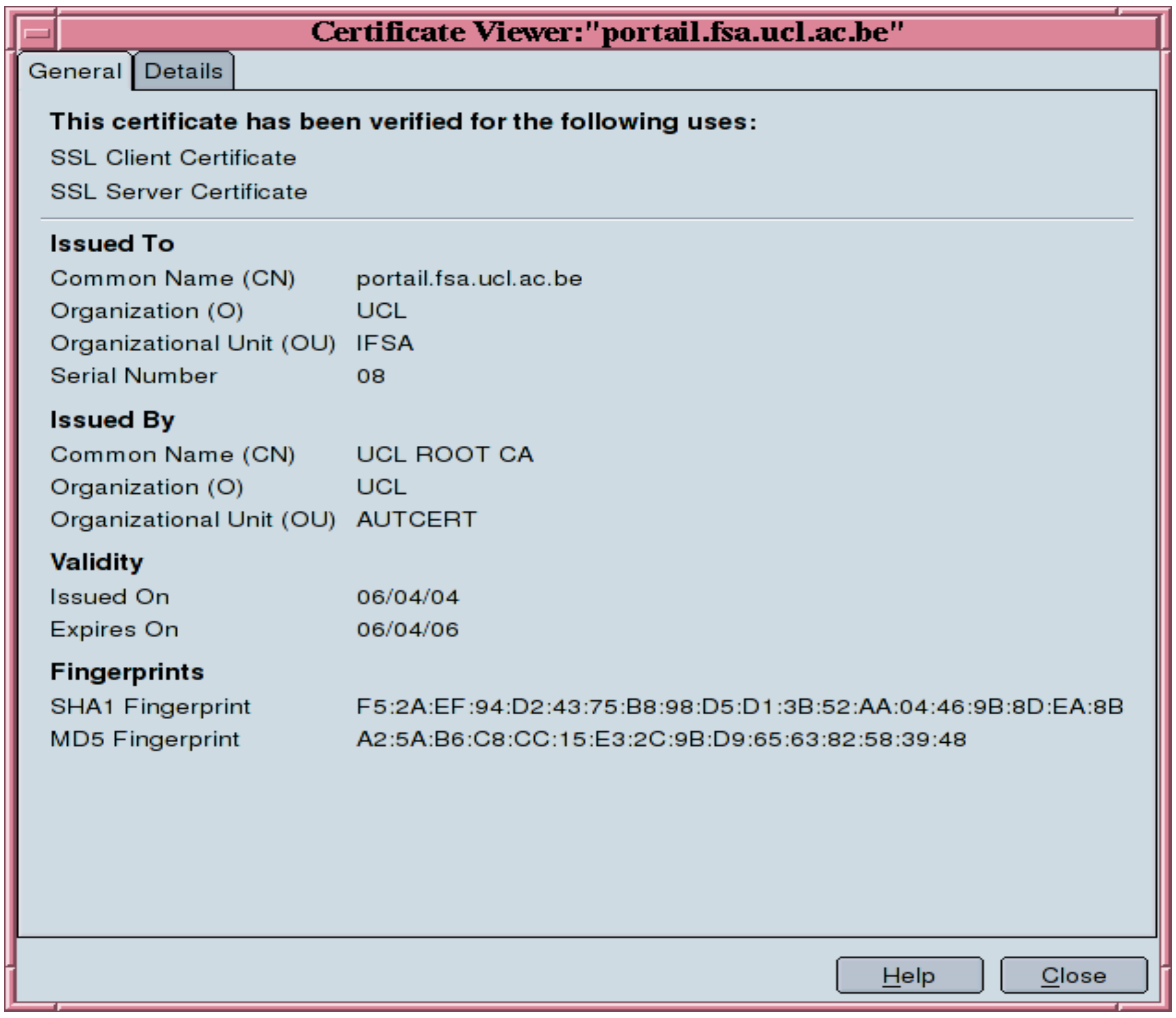
X.509 certificates

- A standard method to encode certificates
  - defined before the creation of SSL
  - intended to be used by OSI applications and encoded in ASN.1
- Example
  - signature
    - algorithm : md5withRSAEncryption
  - Issuer
    - C=US, O=RSA Data Security, Inc., OU=Secure Server Certification Authority
  - Validity
    - not before : *Date* not after : *Date*
  - Subject
    - C=US, ST=Washington, L=Seattle, O=Amazon.com, Inc., CN=www.amazon.com, public key
  - Signature

The certificates were initially an extension to the X.500 directory service developed for OSI applications. A simplified version of this directory service served as the basis for the LDAP directory built by the IETF. LDAP is used inside some enterprises but there are no global deployments as for the DNS.

Certificates used by web servers

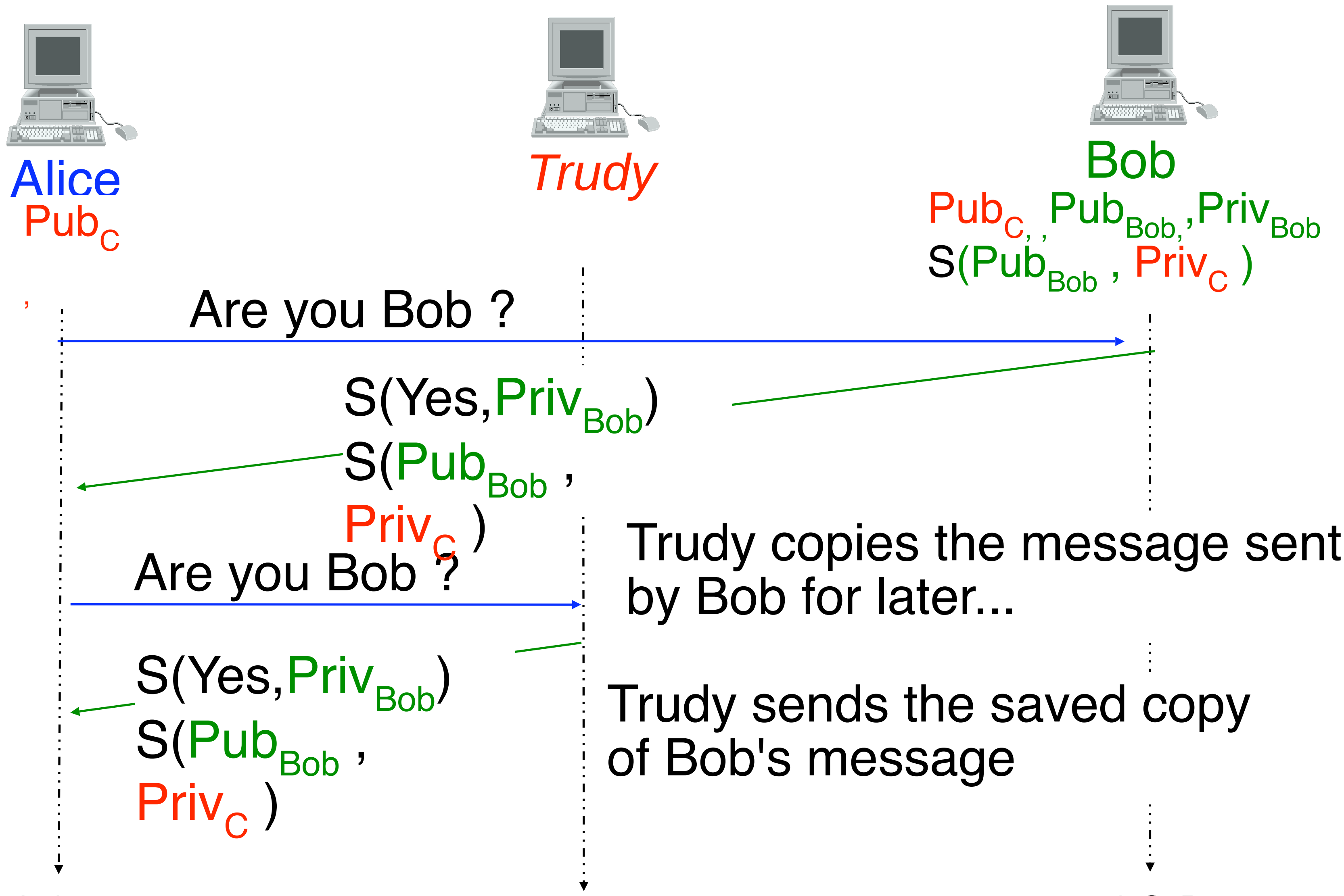
- Example



Example of a CA certificate in the mozilla browser

Are certificates sufficient ?

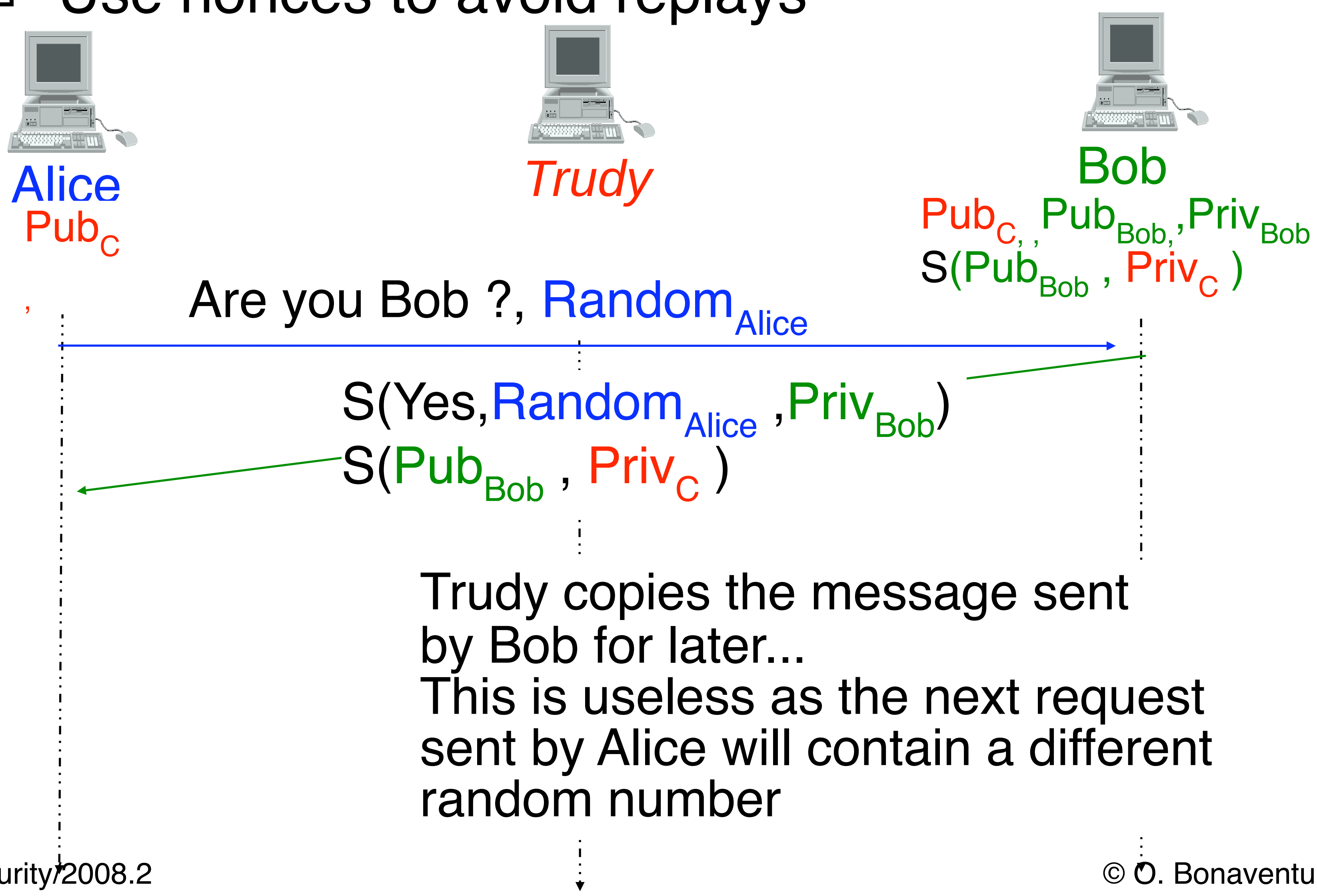
- Replay attacks



Replay attacks are common threats to security protocols.

Are certificates sufficient (2) ?

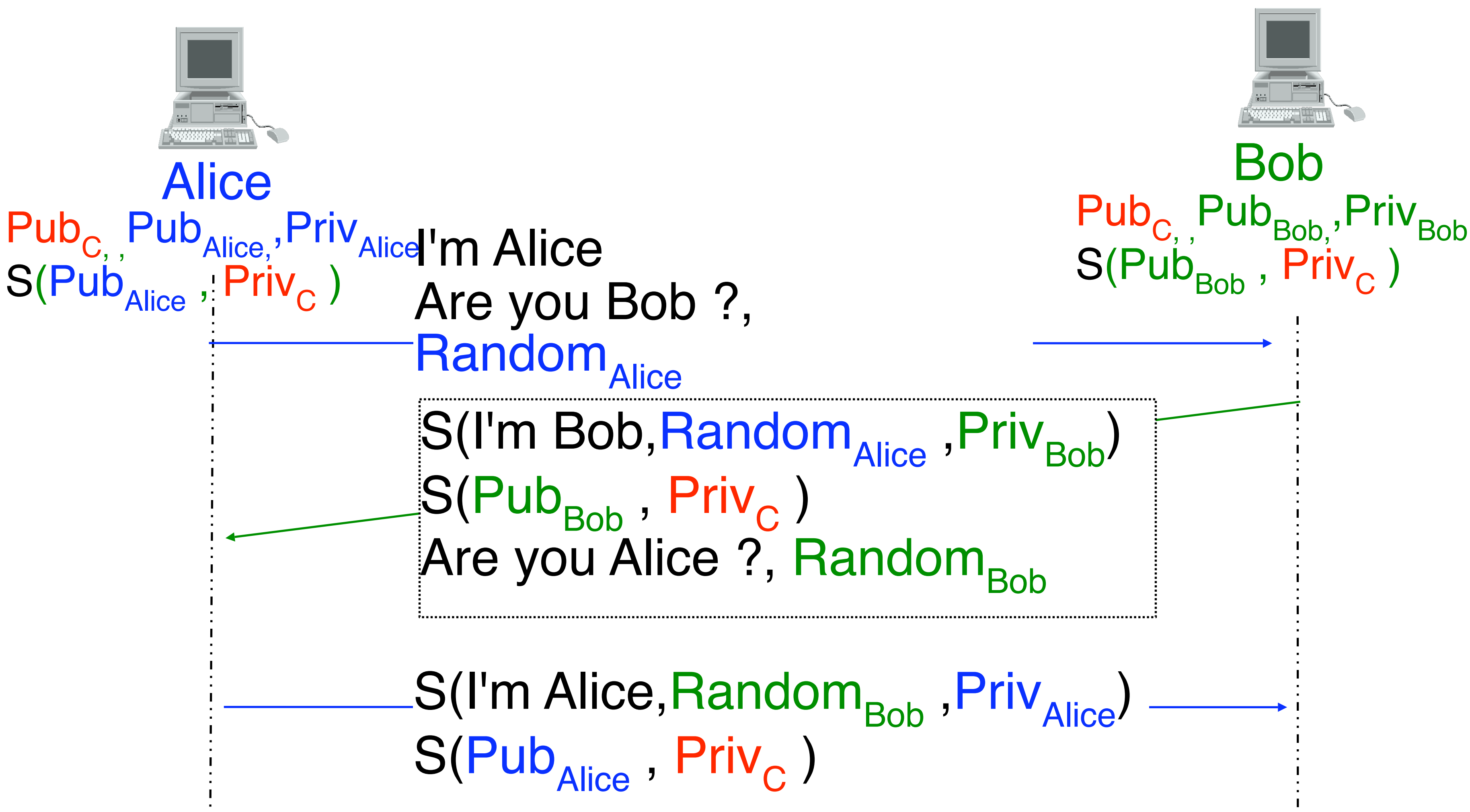
- Solution
  - Use nonces to avoid replays



The nonce is a random number. Note that to be secure, this nonce should be truly random. In practice, generating random numbers is not easy, For detailed discussion, see :
RFC1750 Randomness Recommendations for Security. D. Eastlake, S. Crocker, J. Schiller. December 1994.

Can we authenticate the client ?

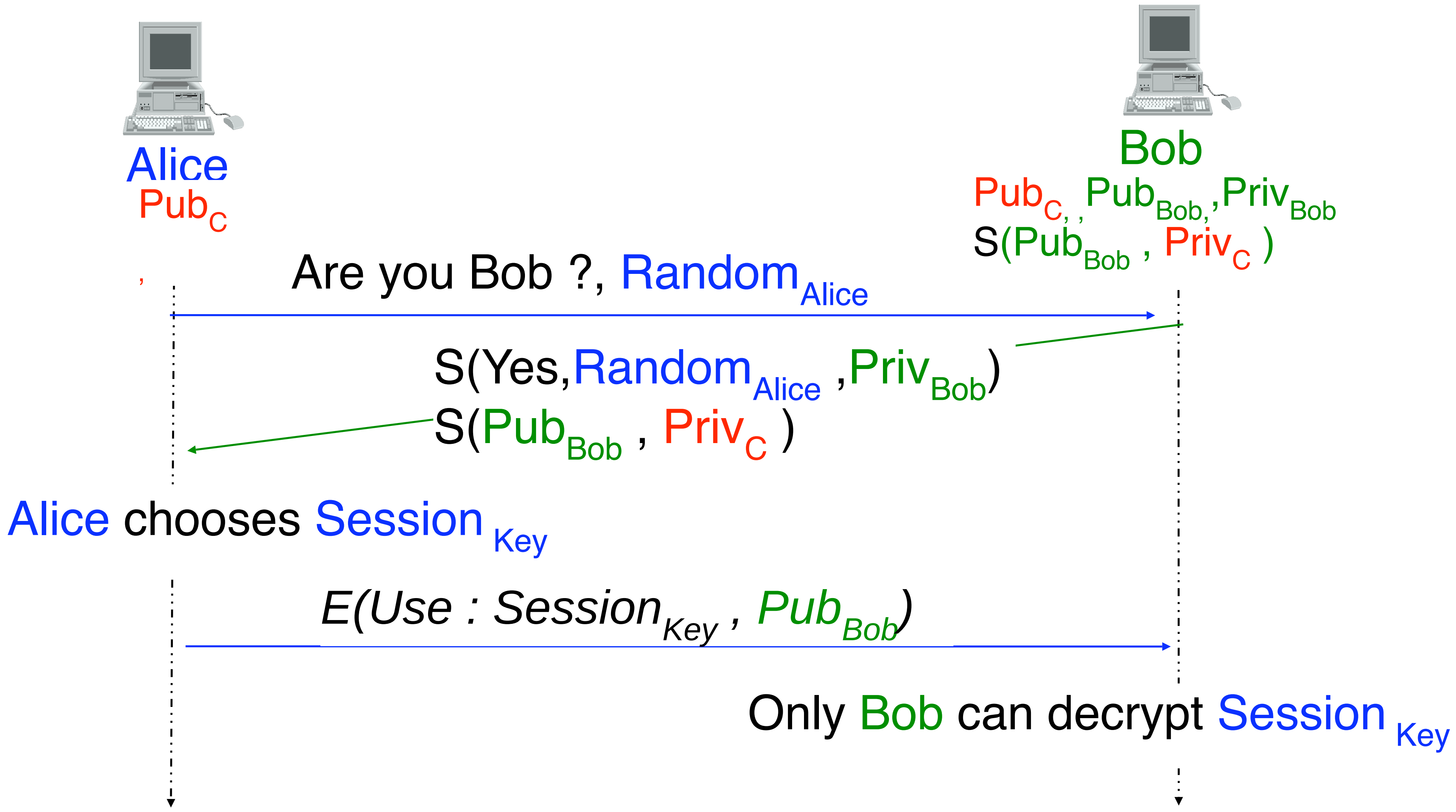
- Principle
  - Use certificates as for the server authentication



Note that in practice, Bob and Alice could know the public key of several trusted third parties in order to check the generated certificates. Only one is shown in the slide for graphical reasons.

How to negotiate an encryption key ?

- Client chooses encryption key



The key chosen by Alice could be a random number. As always, the security of the implementation will depend on the difficulty for an attacker to predict the key that Alice will choose.

How to negotiate an encryption key (2)?

- In practice, data will be sent
  - by client to server
  - by server to client

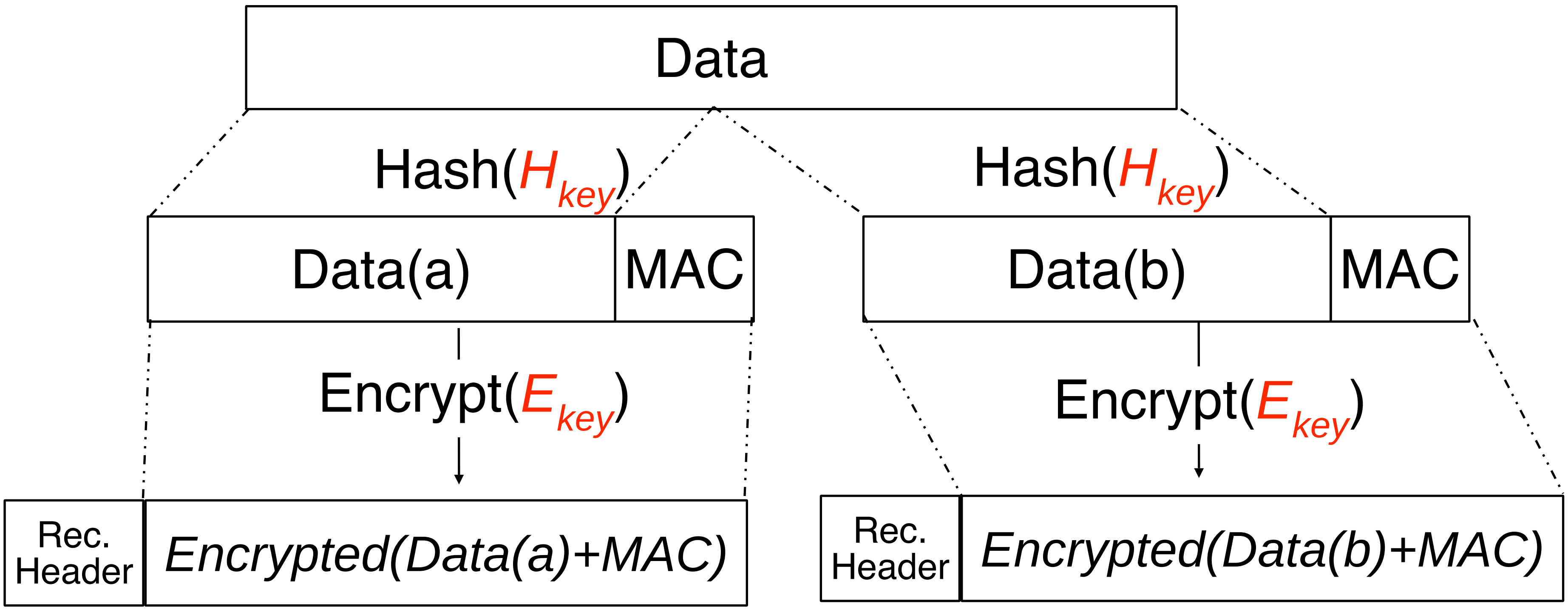
- Using a single key to encrypt two directions is a bad idea since when one key is broken, both directions can be decrypted

- Principle of the solution
  - Alice chooses a PreMasterSecret and uses $Random_{Alice}$ to compute several keys
    - Alice computes the Alice->Bob and Bob->Alice keys
    - Bob computes the Bob->Alice and Alice->Bob keys

Of course, with this scheme Alice and Bob must use the same algorithm to generate the Session keys with the PreMasterSecret. This number should be sent encrypted, e.g. with Bob's public key, to ensure that an attacker cannot capture it.

How to avoid packet injection attacks ?

- Principle
  - TCP offers a byte stream service
  - Divide the byte stream in records
  - Each record is authenticated - and encrypted

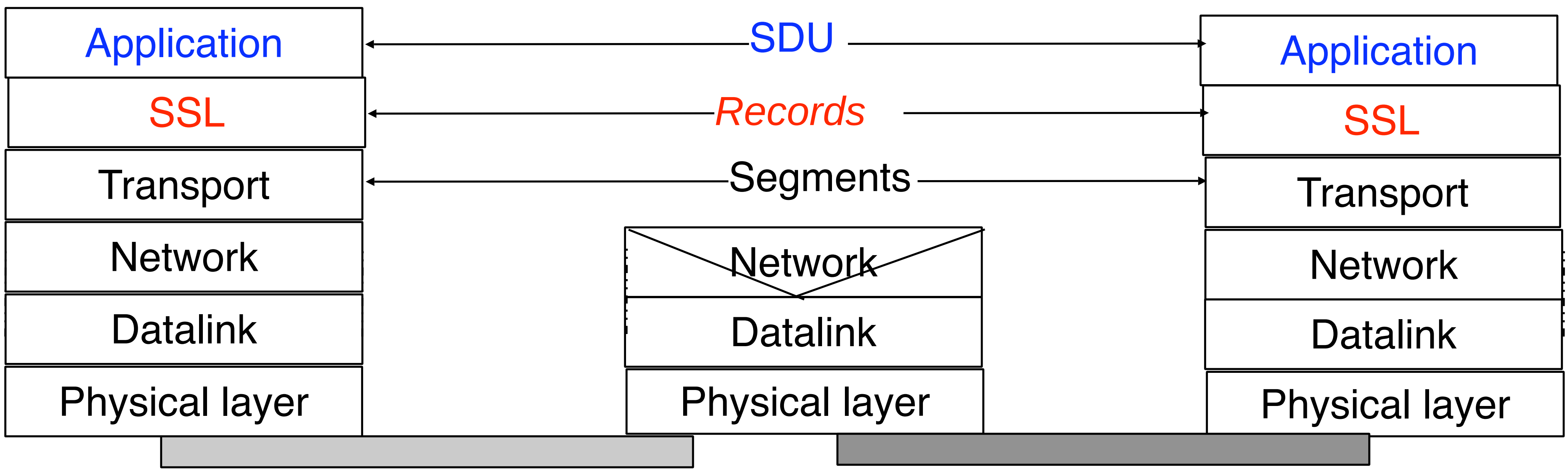


For the same reason as in the previous slide, the encryption and hash keys used for both directions should differ.

The utilization of a MAC inside the records allows to detect packet or record injection attacks. The record header contains information such as the type of record and its length.

Secure Socket Layer

- Principle
  - Add an authentication and encryption layer between the application and transport layers



This section is partially based on

C. Kaufman, R. Perlman, M. Speciner, Network Security : Private communication in a Public world, Prentice Hall

and

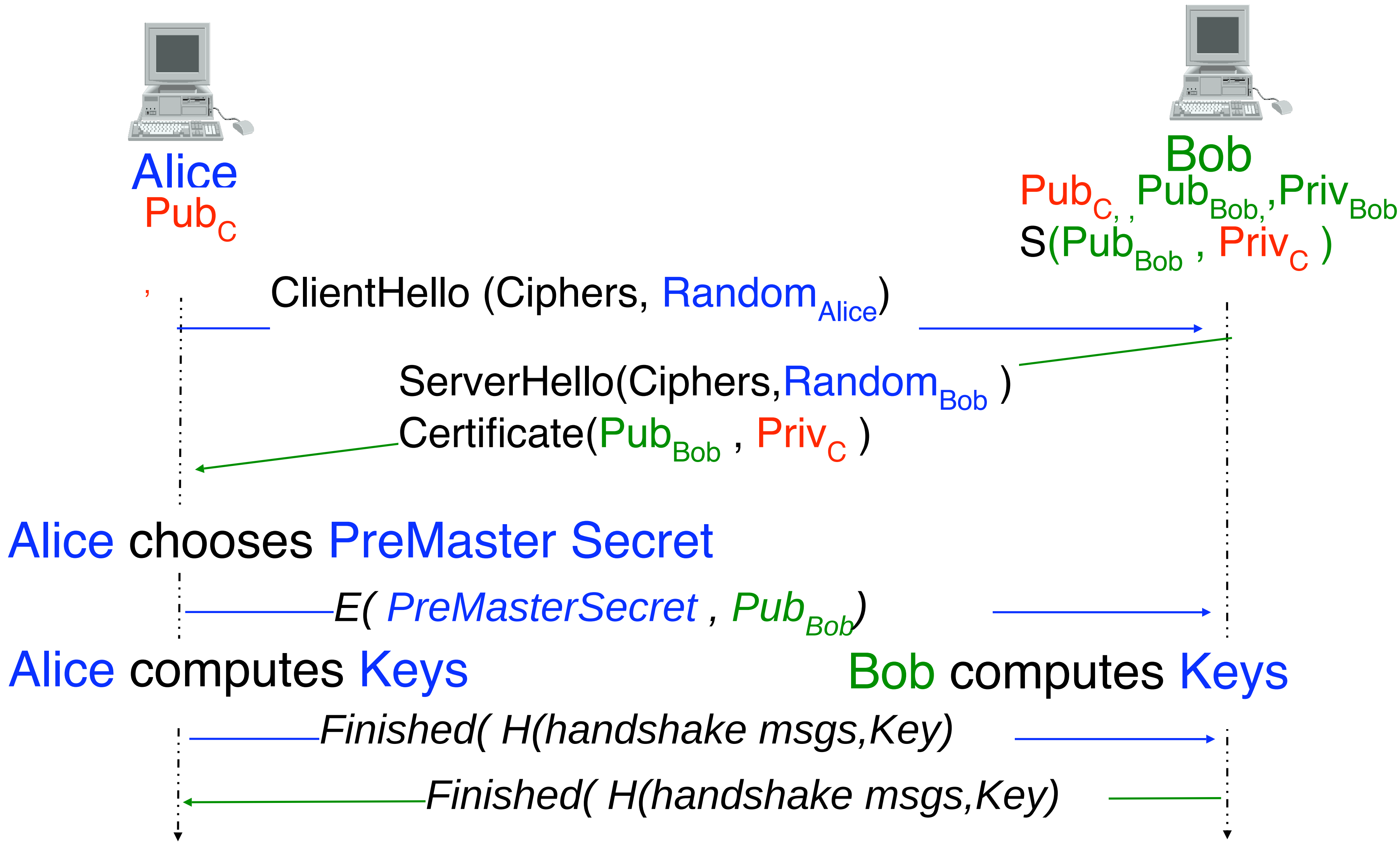
R. Rescorla, SSL and TLS: Designing and Building Secure Systems, Addison Wesley, 2001

Phases of an SSL session

- Handshake phase
  - Session establishment
  - Messages are sent non-encrypted
  - Last messages authenticate the exchange
- Data transfer phase
  - Encrypted and authenticated records are exchanged
  - used to perform real data transfer
- Session termination
  - Data transfer stops and session terminates

Handshake messages

- Principle of SSL session establishment



Each SSL message is encoded as a variable length Type, Length, Value triple. The following types of handshake messages are defined :
HelloRequest
ClientHello
ServerHello
Certificate
ServerKeyExchange
CertificateRequest
ServerHelloDone
CertificateVerify
ClientKeyExchange
Finished

Handshake messages ClientHello

- Used by the Client to initiate SSL session
  - sent in clear without signature
- Contents
  - Protocol Version
    - There are several variants of the SSL specification
  - 32 bytes long random number
    - Composed of two parts
      - 4 bytes Unix time (number of seconds since 01/01/1970)
      - 28 bytes random number
  - Session Id
    - Optional
      - Used by client to resume a previous SSL session
      - Each SSL session has an identifier which can be used later to restart a session
  - List of supported Ciphers
  - List of supported Compression Methods

The main variants of the SSL specification are :
SSLv2 defined in
K. Hickman, The SSL Protocol, Feb. 1995
http://wp.netscape.com/eng/security/SSL_2.html

SSLv3 defined in

A. Freier, P. Karlton, P. Kocher, The SSL Protocol, version 3.0, Internet draft, draft-freier-ssl-version3-02.txt, work in progress, Nov. 1996

TLS defined in
T. Dierks, C. Allen, The TLS protocol, version 1.0, RFC2246, Jan 1999

Due to patent issues, the standardization bodies took a long time before defining compression methods to be used with SSL/TLS.

S. Hollenbeck,, Transport Layer Security Protocol Compression Methods, RFC3749, May 2004

Recently, LZS was added :

R. Friend, Transport Layer Security (TLS) Protocol Compression Using Lempel-Ziv-Stac (LZS), RFC3943, Nov. 2004

Handshake messages ClientHello (2)

- List of supported ciphers
  - In fact a list (authentication + key exchange + cipher + hash)
    - Authentication
      - RSA or DSS
    - Key Exchange
      - RSA, Diffie Hellman
    - Encryption
      - None, RC4(40 bits), RC4 (128 bits), DES, 3DES, IDEA
    - Hash
      - SHA or MD5
  - Some combinations are stronger than others
    - Example
      - TLS RSA WITH NULL MD5
      - TLS RSA EXPORT WITH RC4 40 MD5
      - TLS RSA WITH RC4 128 MD5
      - TLS RSA WITH DES CBC SHA
      - TLS RSA WITH 3DES EDE CBC SHA

For example, here are the ciphers supported by openssl, a freely available SSL library :

```
DHE-RSA-AES256-SHA      SSLv3 Kx=DH       Au=RSA  Enc=AES(256)  Mac=SHA1
DHE-DSS-AES256-SHA      SSLv3 Kx=DH       Au=DSS  Enc=AES(256)  Mac=SHA1
AES256-SHA              SSLv3 Kx=RSA      Au=RSA  Enc=AES(256)  Mac=SHA1
EDH-RSA-DES-CBC3-SHA    SSLv3 Kx=DH       Au=RSA  Enc=3DES(168) Mac=SHA1
EDH-DSS-DES-CBC3-SHA    SSLv3 Kx=DH       Au=DSS  Enc=3DES(168) Mac=SHA1
DES-CBC3-SHA            SSLv3 Kx=RSA      Au=RSA  Enc=3DES(168) Mac=SHA1
DES-CBC3-MD5            SSLv2 Kx=RSA      Au=RSA  Enc=3DES(168) Mac=MD5
DHE-RSA-AES128-SHA      SSLv3 Kx=DH       Au=RSA  Enc=AES(128)  Mac=SHA1
DHE-DSS-AES128-SHA      SSLv3 Kx=DH       Au=DSS  Enc=AES(128)  Mac=SHA1
AES128-SHA              SSLv3 Kx=RSA      Au=RSA  Enc=AES(128)  Mac=SHA1
RC2-CBC-MD5             SSLv2 Kx=RSA      Au=RSA  Enc=RC2(128)  Mac=MD5
DHE-DSS-RC4-SHA         SSLv3 Kx=DH       Au=DSS  Enc=RC4(128)  Mac=SHA1
RC4-SHA                 SSLv3 Kx=RSA      Au=RSA  Enc=RC4(128)  Mac=SHA1
RC4-MD5                 SSLv3 Kx=RSA      Au=RSA  Enc=RC4(128)  Mac=MD5
RC4-MD5                 SSLv2 Kx=RSA      Au=RSA  Enc=RC4(128)  Mac=MD5
RC4-64-MD5              SSLv2 Kx=RSA      Au=RSA  Enc=RC4(64)   Mac=MD5
EXP1024-DHE-DSS-DES-CBC-SHA SSLv3 Kx=DH(1024) Au=DSS  Enc=DES(56)   Mac=SHA1 export
EXP1024-DES-CBC-SHA     SSLv3 Kx=RSA(1024) Au=RSA  Enc=DES(56)   Mac=SHA1 export
EXP1024-RC2-CBC-MD5     SSLv3 Kx=RSA(1024) Au=RSA  Enc=RC2(56)   Mac=MD5  export
EDH-RSA-DES-CBC-SHA     SSLv3 Kx=DH       Au=RSA  Enc=DES(56)   Mac=SHA1
EDH-DSS-DES-CBC-SHA     SSLv3 Kx=DH       Au=DSS  Enc=DES(56)   Mac=SHA1
DES-CBC-SHA             SSLv3 Kx=RSA      Au=RSA  Enc=DES(56)   Mac=SHA1
DES-CBC-MD5             SSLv2 Kx=RSA      Au=RSA  Enc=DES(56)   Mac=MD5
EXP1024-DHE-DSS-RC4-SHA SSLv3 Kx=DH(1024) Au=DSS  Enc=RC4(56)   Mac=SHA1 export
EXP1024-RC4-SHA         SSLv3 Kx=RSA(1024) Au=RSA  Enc=RC4(56)   Mac=SHA1 export
EXP1024-RC4-MD5         SSLv3 Kx=RSA(1024) Au=RSA  Enc=RC4(56)   Mac=MD5  export
EXP-EDH-RSA-DES-CBC-SHA SSLv3 Kx=DH(512)  Au=RSA  Enc=DES(40)   Mac=SHA1 export
EXP-EDH-DSS-DES-CBC-SHA SSLv3 Kx=DH(512)  Au=DSS  Enc=DES(40)   Mac=SHA1 export
EXP-DES-CBC-SHA         SSLv3 Kx=RSA(512) Au=RSA  Enc=DES(40)   Mac=SHA1 export
EXP-RC2-CBC-MD5         SSLv3 Kx=RSA(512) Au=RSA  Enc=RC2(40)   Mac=MD5  export
EXP-RC2-CBC-MD5         SSLv2 Kx=RSA(512) Au=RSA  Enc=RC2(40)   Mac=MD5  export
EXP-RC4-MD5             SSLv3 Kx=RSA(512) Au=RSA  Enc=RC4(40)   Mac=MD5  export
EXP-RC4-MD5             SSLv2 Kx=RSA(512) Au=RSA  Enc=RC4(40)   Mac=MD5  export
```

Handshake messages ServerHello

- Used by the Server to reply to ClientHello
  - Sent in clear without signature

- Contents
  - Protocol version
    - Highest version of the protocol supported by both client and server
  - Random
    - A random structure generated by the server
  - Session Id
    - Optional, sent by server if it allows sessions to be resumed later
  - Cipher Suite
    - One of the cipher suites proposed by the client
  - Compression Method

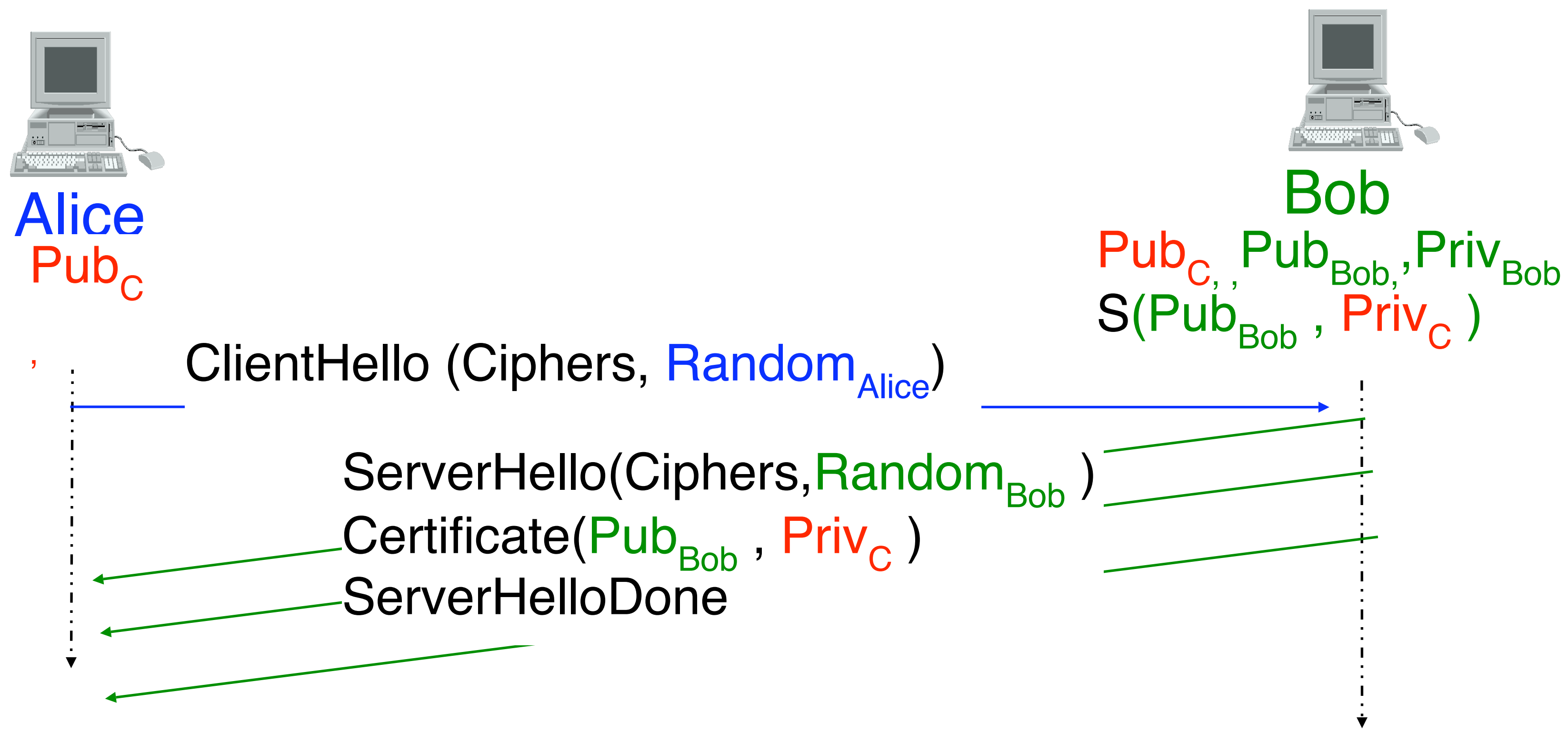
The session id indicates the identifier of the SSL session. Servers and Clients may cache session information to be able to resume those sessions later. This is particularly useful for application protocols such as HTTP 1.0 where several TCP connections are established between the client and the server.

Handshake messages
Certificate

- Utilisation of the Certificate message
  - Contains a list of X.509 certificates and is sent in plain
    - server certificate
    - certificates of certification authorities if any
    - certificates are encoded in ASN.1
  - Sent by the server to authenticate itself
    - a server may have several certificates from different certification authorities
  - Certificate can also be sent by the client when client authentication is requested by the server with CertificateRequest

Handshake messages ServerHelloDone

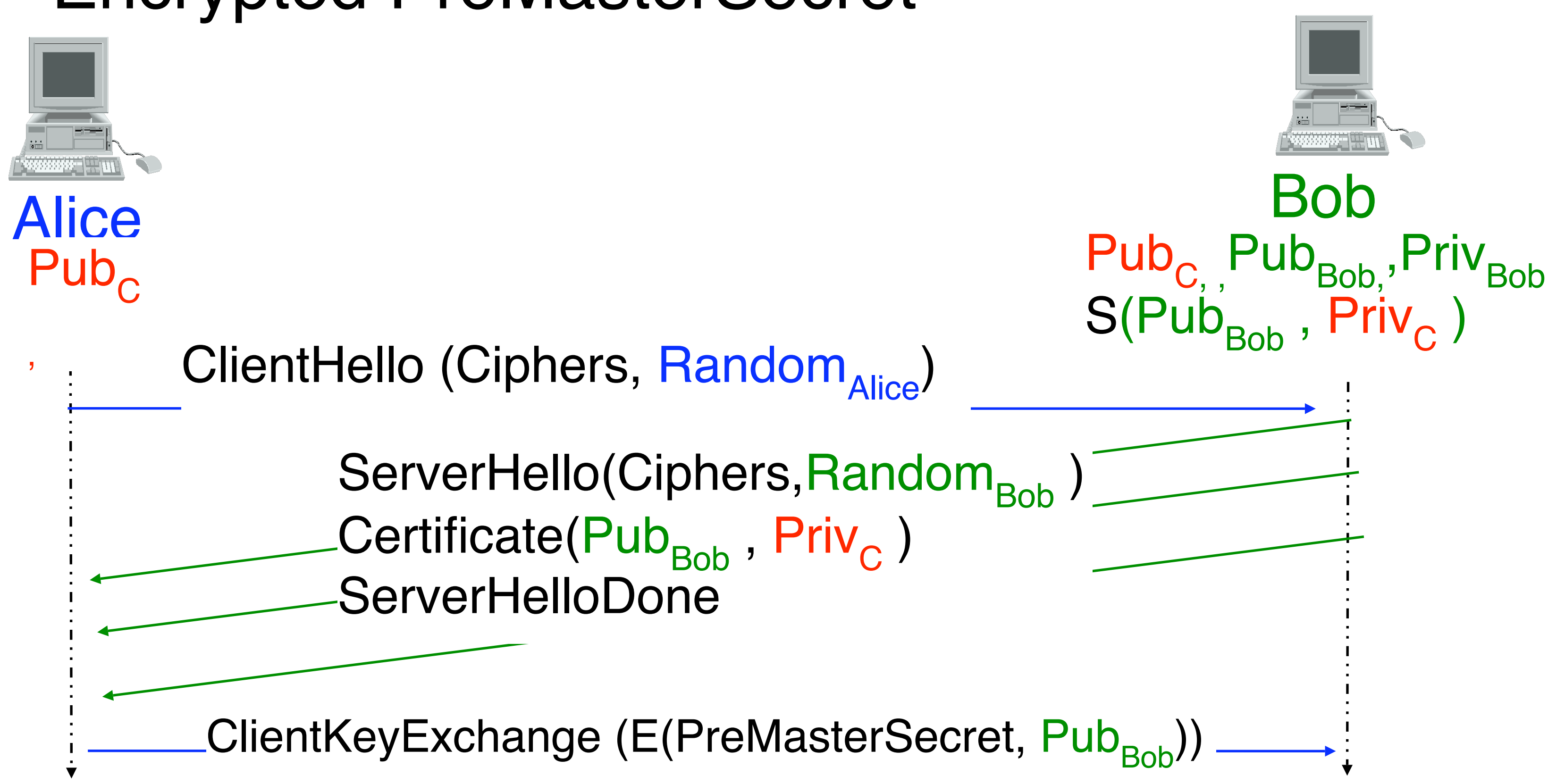
- Utilisation
  - Indicates that the server has finished its first phase of the handshake
  - sent in plain unencrypted



The ServerHelloDone is a simple message to indicate that the server has sent all the required information to establish the SSL session. It does not contain any parameter.

Handshake messages ClientKeyExchange

- Utilisation
  - used by the client to send the PreMasterSecret encrypted with the server's public key
- Contents
  - Encrypted PreMasterSecret

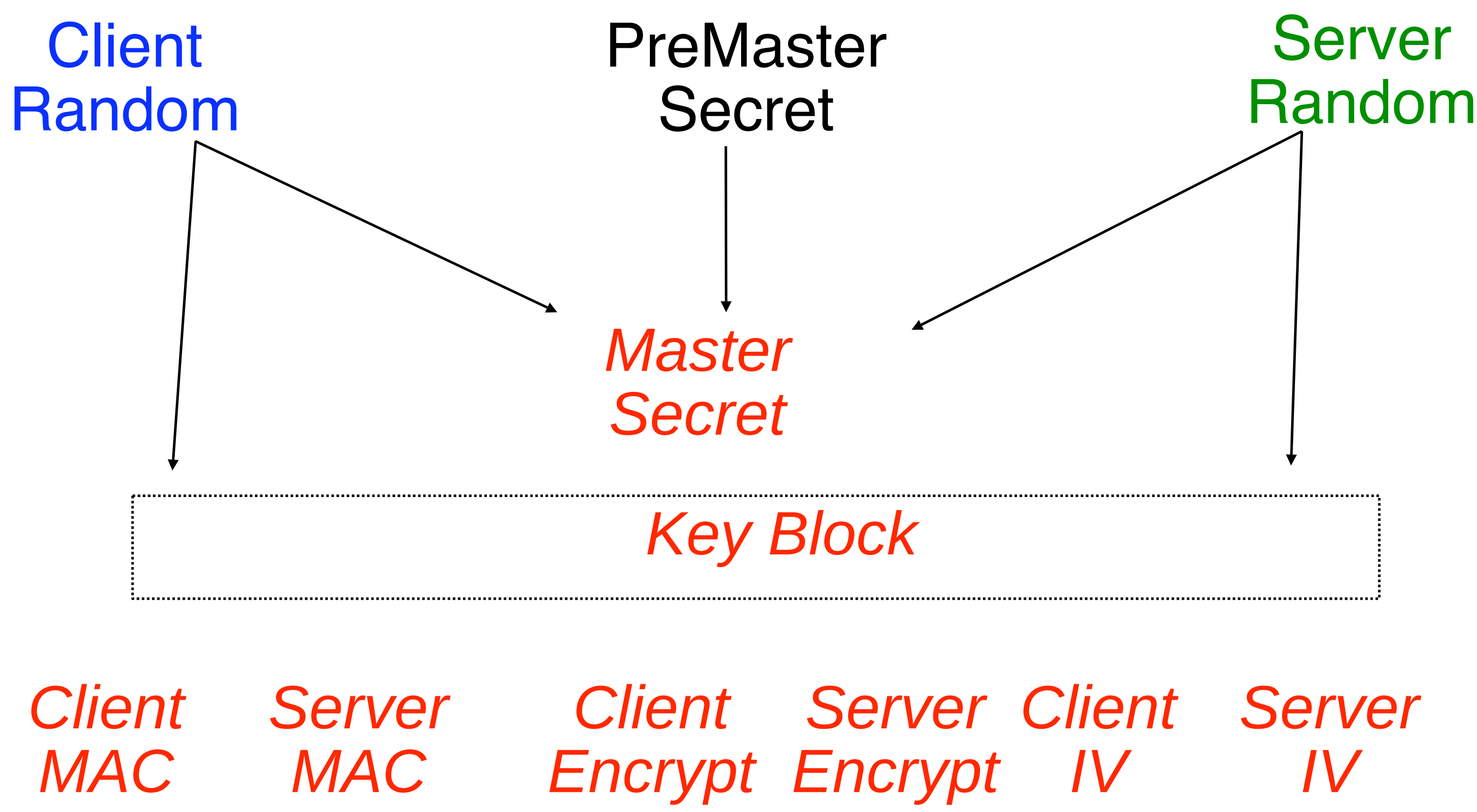


The ClientKeyExchange message is the only message that contains information encrypted with the server's public key.

The PreMasterSecret is used by the server and the client to compute the secret keys are necessary to encrypt and authenticate the data records exchanged over the SSL session.

Key derivation

- Principle



Most secret-key encryption algorithms work on a block basic. They encrypt (or decrypt) a block of 8 bytes or 16 bytes of data based on the value the secret key. To use those algorithms, the data must obviously be divided in blocks of bytes.

A first solution to encrypt those blocks is to consider that each block is independent. This is often called the Electronic Codebook (ECB) mode. In this case, the i^{th} encrypted block is E(M[i], k) where M[i] is the i^{th} message block. A drawback of this method is that two identical blocks of tl message will appear as the same encrypted block. This could reveal information about the message to an attacker. For this reason, most block-base encryption schemes are used in Cipher Block Chaining (CBC) mode. In this mode, the i^{th} encrypted block is a function of both the i^{th} message blo and the i-1 encrypted block.
C[i] = E(M[i] XOR C[i-1] , k)
To use CBC mode, we need to define how the first message block will be encrypted. This is done by using an Initialization vector (IV) that is used a C[-1]. The IV is computed by the client and the server.

Key derivation in SSLv3

- Principle
  - Use both MD5 and SHA-1 to derive keys
  - Computation of MasterSecret
    - MD5(PreMasterSecret + SHA-1("A" + PreMasterSecret + $Client_{Random}$ + $Server_{Random}$))+ MD5(PreMasterSecret + SHA-1("BB" + PreMasterSecret + $Client_{Random}$ + $Server_{Random}$))+ MD5(PreMasterSecret + SHA-1("CCC" + PreMasterSecret + $Client_{Random}$ + $Server_{Random}$))
  - Computation of Key Block
    - MD5(MasterSecret + SHA-1("A" + MasterSecret + $Client_{Random}$ + $Server_{Random}$))+ MD5(MasterSecret + SHA-1("BB" + MasterSecret + $Client_{Random}$ + $Server_{Random}$))+...

Both the client and the server know all the information required to compute the Key block.

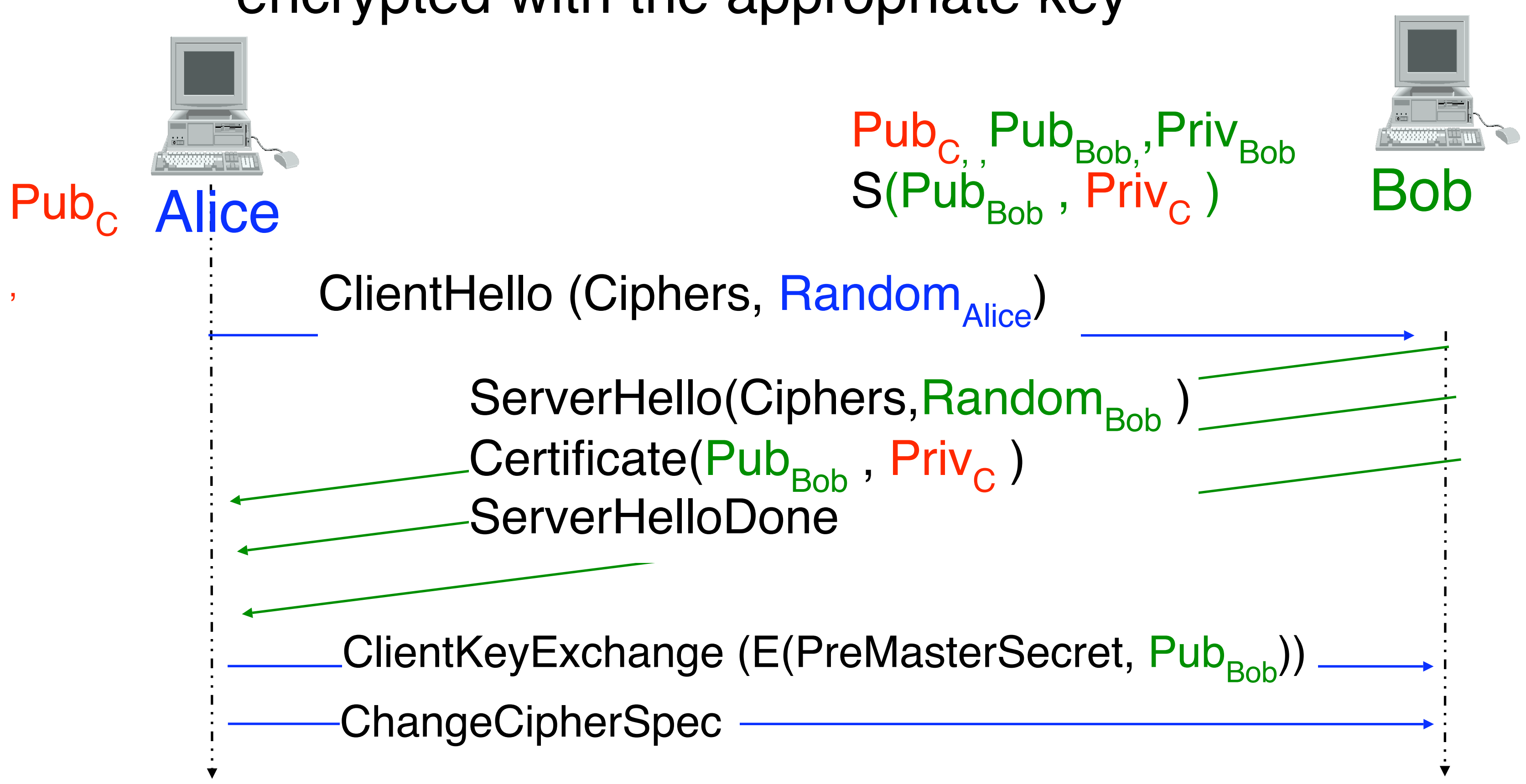
The computation of the key block uses as many round as required to provide enough bits for the key block depending on the type of encryption scheme used. The key block is then divided in six parts to obtain the MAC keys, the encryption keys and the IV's. When exportable ciphers are used the generated keys must be weakened.

The utilisation of both MD5 and SHA-1 was a design choice to reduce the risk that a weakness found in one hash function could be used to attack the key derivation function.

The computation of the keys is slightly more complex in TLS, but the principle is the same.

Handshake messages ChangeCipherSpec

- Utilisation
  - Used by client and server to indicate that they start using a (new) key
  - During handshake, indicates that next message will be encrypted with the appropriate key



Handshake messages Finished

- Utilisation
  - Sent by both client and server to confirm the establishment of the secure SSL session
    - Session is established only is client received expected Finished message from server and vice-versa
  - Allows to detect man in the middle attacks on ClientHello and ServerHello messages
    - example
      - Attacker changes cipher list to propose weaker ciphers
  - First encrypted message on each direction
- Contents
  - Keyed hash (MD5 or SHA-1) of all the handshake messages and the MasterSecret

The keyed hash found in the Finished message is computed in SSLv3 as follows :

hash=MD5(MasterSecret + pad2 +
MD5 (Handshake messages + *Sender* + MasterSecret + pad1))

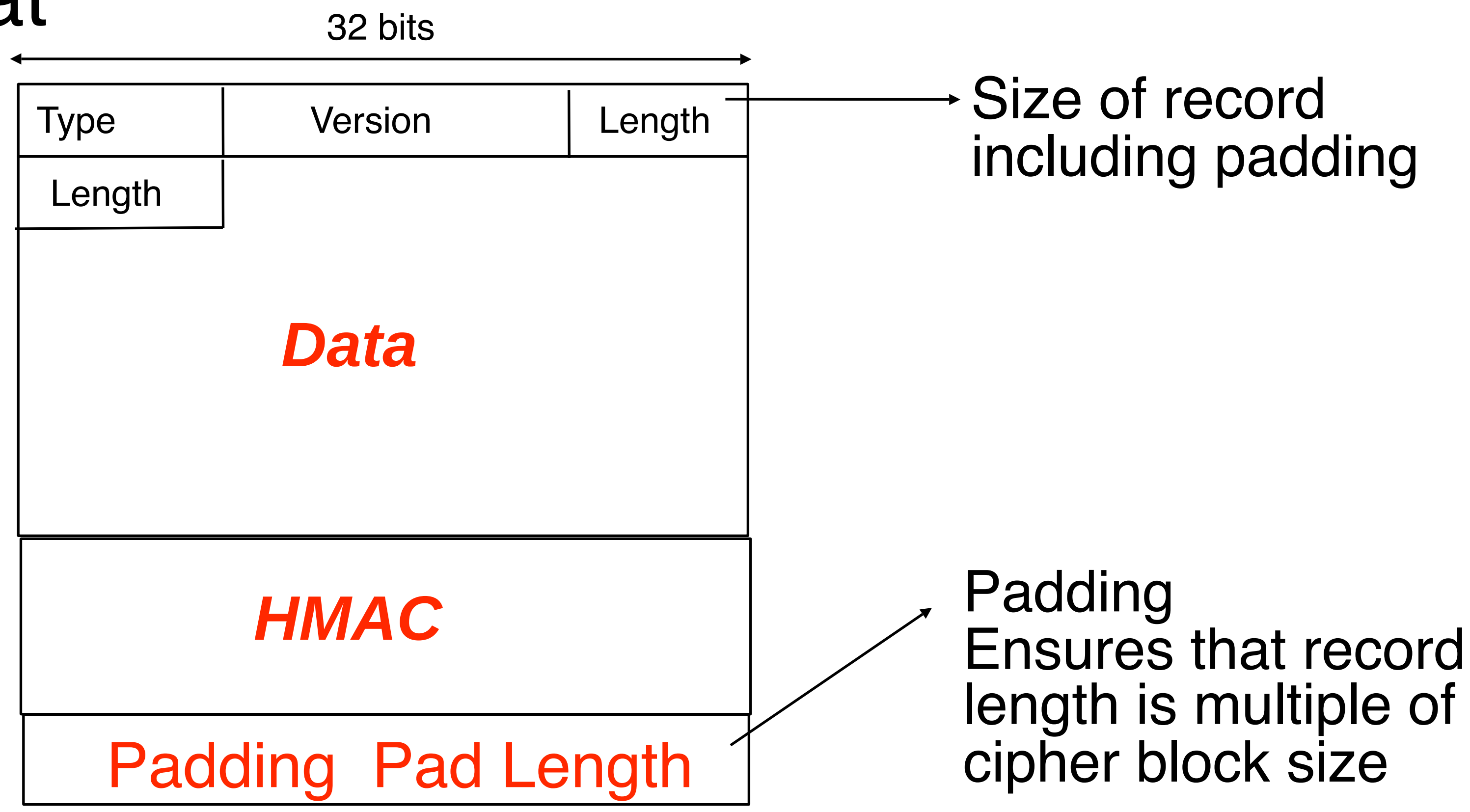
In this function, Sender is a constant set to 0x434C4E54 on the client and 0X53525652 on the server. This ensures that the hash computed by the server will differ from the hash computed by the client to avoid replay attacks.
pad1 is a string of byte 0x36 repeated 48 times and pad2 0x5c repeated 48 times.

MD5 can be replaced by SHA-1 when this hash has been selected.

The computation of the key hash in TLS is slightly different.

SSL records

- Utilisation
  - Transmission of encrypted and authenticated user data
- Format



The maximum size of a SSL record is 2^{14} -1 bytes

The type, version and length fields of the SSL record are sent in plain, unencrypted. The other parts of the record are encrypted by using the write key.

Authentication of SSL records

- Computation of HMAC
  - TLS
    - MAC = hash($Send_{hash}$ + Seq_{num} + type + version + length + data)
  - SSLv3
    - hash($Send_{hash}$+ pad2 +
      hash($Send_{hash}$+ pad1 +Seq_{num} + length + data))

- Parameters
  - $Send_{hash}$ derived from Key Block
  - Seq_{num} 64 bits sequence number used to detect replay and reordering attacks
    - note that when the received hash does not match, there is no retransmission mechanism in SSL

For a more detailed presentation of the HMAC algorithm used by SSL to compute the message digest for the SSL records, see :

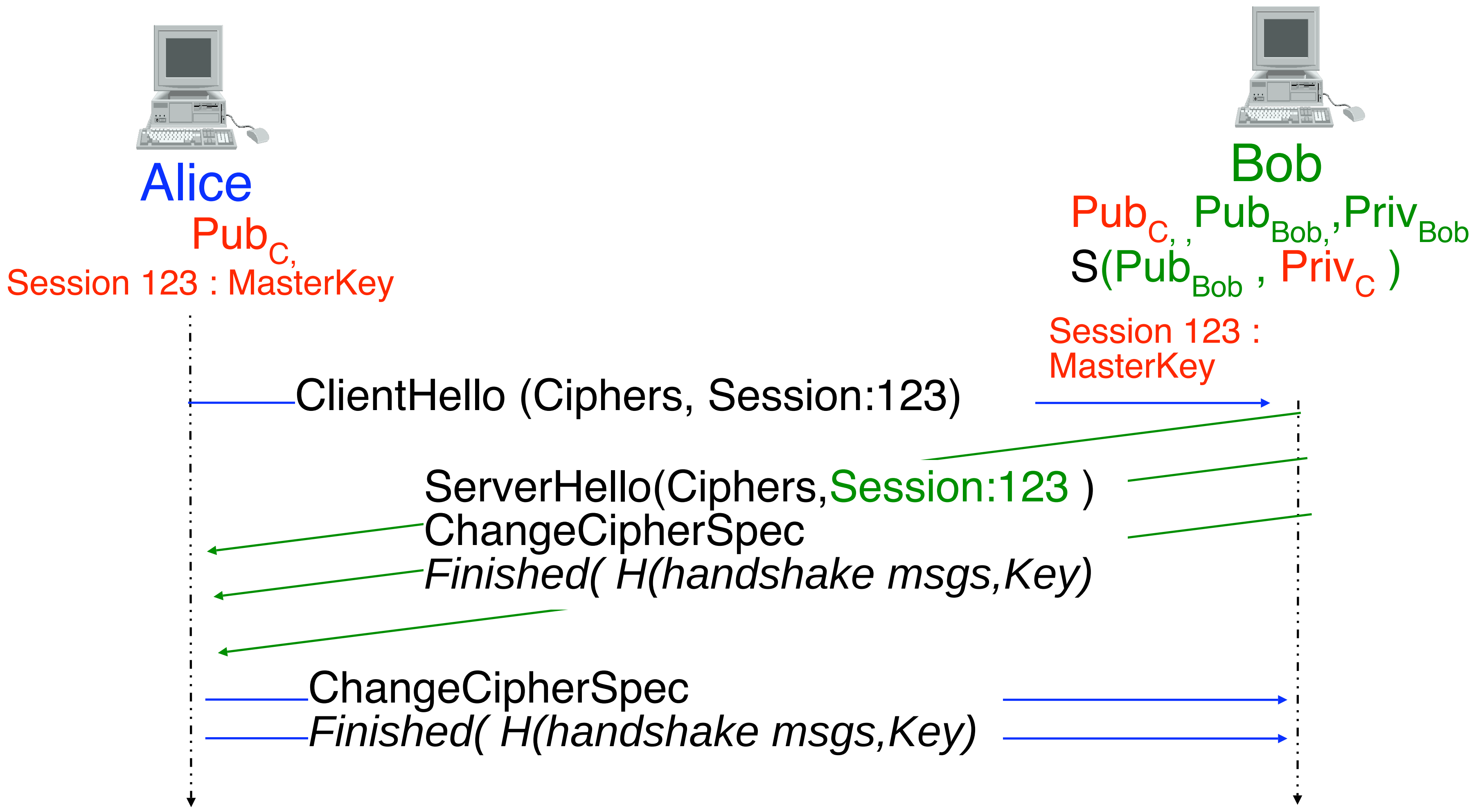
H. Krawczyk, M. Bellare, R. Canetti, HMAC : keyed hashing for message authentication, RFC2104, Feb 1997

SSL alerts

- Messages used to inform of problems on a SSL session
- Examples
  - bad_record_mac
    - a record with bad MAC was received, session closed
  - handshake failure
    - failure during the establishment of the SSL session
  - bad_certificate
    - certificate was corrupted or invalid
  - revoked certificate / certificate expired
    - certificate is not valid anymore
  - unknown ca
    - certificate was singed by an unknown cert. authority
  - insufficient security
    - ciphers proposed are not secure enough

SSL session resumption

- Principle
  - Client/Server cache SessionId and MasterSecret



The main advantage of resuming previous SSL sessions is that this allows to avoid recomputing the MasterKey and sending it encrypted. This can speed up the establishment of the SSL session given the cost of performing public key encryption and decryption.

If the server does not agree to resume the session, then it simply generates a new session id and places it in the ServerHello message.

On most implementations, session resumption is possible even if the client uses a different IP address and different ports numbers. Using a different port number is normal given how TCP ports are allocated on most operating systems. Using a different IP address may be normal for mobile clients or clients that are using DHCP. The validation of the SSL session is based on the ability to compute the Finished message which is independent or the IP addresses and port numbers.

SSL client authentication

- Principle
  - Server requires client to provide a valid certificate to agree to establish session

- New messages
  - CertificateRequest
    - Sent by server to request client certificate
    - Contains certificate type and list of acceptable certification authorities

  - CertificateVerify
    - Sent by client to prove to the server that it knows the private key of the certificate that it sent
    - Content
      - Signature of all the handshake messages sent and received with the client private key

The CertificateRequest message contains the list of certification authorities that are considered as valid by the server. The client must provide a certificate issues by one of those certification authorities otherwise the server will not agree to establish the SSL session

The CertificateVerify is necessary to allow the server to verify that the client is able to encrypt something with the private key associated to the certified public key. As the client signs the handshake messages, it also signs the random number chosen by the server. This avoids replay attacks.

With the CertifcateVerify message, there is some asymmetry between the server and the client. The client uses the CertificateVerify message to prove that it knows the key announced in the certificate. The server does not send such a message. This is not necessary as the server must know the private key corresponding to its certificate to decrypt the ClientKeyExchange message and correctly compute the session keys and thus the Finished message.

Ephemeral keys

- Problem
  - When SSL was designed, long RSA keys could not be used with export clients
- Solutions
  - Each server maintains a long and a short key
    - server must maintain several certificates
    - operational issues on server
  - Ephemeral key
    - Server generates random short key for each session
      - short key can be broken by government agencies if required
    - short key is signed by using the long server key
      - ensures that
    - client validates the short key's signature and use it to encrypt the PreMasterSecret

Security issues with SSL

- Master secret must remain secret
- Server's private must remain secret
- Random number generators
- Certificates should be checked
- Cipher negotiation

Security of MasterSecret

- Computed by client and server based on *PreMasterSecret*, Random $_{Client}$, $Random_{Server}$
- Security risk
  - If attacker knows MasterSecret, he can read all data and inject new data in SSL session
- Storage
  - SSL is usually implemented in software
    - MasterSecret is usually stored in memory
      - on a multi-user machine, a process with administrator rights can read at any memory location
    - MasterSecret should not be stored on disk
      - implementation should make sure that memory containing MasterSecret is locked
      - Core dumps may reveal MasterSecret as well

On Unix, `mlock` can be used to mark memory zones that should not be placed on disk.

Security of private keys

- Problem
  - Server maintains private,public key pair
  - Certified client also uses key pair
- Security risks
  - If server's private key is compromised, then all captured sessions with the server can be recovered
  - If client's private key is compromised, then any other client can impersonate it
- How to protect private keys ?

Protection of client's private key

- Principle
  - User selects a pass phrase
    - pass phrase is much longer than a password
    - pass phrase should be longer than protected key
    - dictionary attacks against pass phrases are more difficult if pass phrase is well chosen
  - Private key is encrypted with pass phrase
- Computation of encrypted key
  - $k = H(H(H(H(H(Pass_{phrase}, Salt)))))$
  - $Encrypted_{Key} = E(Private_{Key}, k)$

    - Salt is stored in plain with the encrypted private key
      - makes dictionary attacks more difficult for attacker
    - H is computed several times to slow brute force and dictionary attacks

Other approaches

- Pass-phrase based private key
  - Principle
    - To generate a key pair, a random number generator is used
      - usually RNG is seeded with with a random seed
      - instead, use the pass phrase to seed the RNG
- Private key stored on hardware
  - dumb device that simply stores the private key
    - PIN number, password or pass phrase used to unlock the private key
  - intelligent device such as a smart card
    - contains key pair, certificate and is able to encrypt
    - software interacts with smart card when message must be encrypted with private key

Protection of the server's private key

- Software-based solutions
  - Private key is protected by OS permissions
  - Private key is encrypted with pass phrase
    - in this case, the administrator must provide pass phrase at each reboot
  - Private key is not encrypted
    - server can automatically reboot, but an attack on the server can reveal the private key
- Hardware-based solutions
  - simple storage device
    - no added security, pass phrase required
  - hardware providing encryption
    - tamper resistant device stores key and encrypts
      - improves performance as well
    - can be protected with a password or pass phrase
    - if device is physically stolen, private key also

Random number generators

- How to obtain good random numbers ?

- Use random physical processes
  - lower bits of counter that counts number of radioactive particles per unit of time
  - thermal fluctuations of electrons wandering through a resistor or a semiconductor junction
    - included in some CPUs like Pentium
- Use pseudo random number generators
  - algorithms that generate a stream of pseudo random numbers
  - stream depends on seed provided
    - most OSes provide today random values to seed the PRNG, by measuring random delays such as time between key presses, delays between interrupts, ...

For physical based random number generators, see e.g.
http://www.americanscientist.org/template/AssetDetail/assetid/20829/page/4?&print=yes

Unix variants provide, in addition to the PRNG found in the standard library of all languages, kernel-based random number generators. Those random numbers are usually available via the /dev/random or /dev/urandom devices

Certificate validation

- Content of the X.509 certificates
  - Not initially developed to certify e-commerce servers
  - Multiple optional fields
    - C=country
    - O=organisation
    - OU=Organisation Unit
    - CN=Common Name
      - sometimes used to encode the DNS name for a server
      - certificates do not contain IP addresses
    - ST=State
    - L=City
    - Key usage extensions
      - digitalSignature, keyEncipherment, dataEncipherment, keyCertSign, ...
    - Optional Fields
      - emailAddress, subjectAltName, ...

Example certificates

- Certificates for servers
  - subject=/C=BE/O=UCL/OU=INGI/CN=renoir.info.ucl.ac.be/emailAddress=webmaster@info.ucl.ac.be
    issuer=/C=BE/O=UCL/OU=CA/CN=UCL Certification Manager/emailAddress=ca@ucl.ac.be
  - subject=/C=US/ST=California/L=Mountain View/O=VeriSign, Inc./OU=Production Services/OU=Terms of use at www.verisign.com/rpa (c)00/CN=www.verisign.com
    issuer=/O=VeriSign Trust Network/OU=VeriSign, Inc./OU=VeriSign International Server CA - Class 3/OU=www.verisign.com/CPS Incorp.by Ref. LIABILITY LTD.(c)97 VeriSign
  - subject=/C=BE/CN=www.belgium.be/O=Belgian Federal Government/OU=Federal Public Service/ ST=Brussels/L=Brussels/emailAddress=servicedesk@fedict.be
    issuer=/C=BE/CN=Government CA

- The examples above were collected by using openssl s_client
- on the following https servers :
- https://renoir.info.ucl.ac.be
- https://www.belgium.be
- https://www.verisign.com

Example certificates (2)

- Certificates provided by CAs
  - self-signed certificate
    - subject:/C=BE/O=UCL/OU=CA/CN=UCL Certification Manager/emailAddress=ca@ucl.ac.be
      issuer:/C=BE/O=UCL/OU=CA/CN=UCL Certification Manager/emailAddress=ca@ucl.ac.be
  - certificate chain signed by a root CA
    - subject:/C=BE/O=GlobalSign nv-sa/OU=Root CA/CN=GlobalSign Root CA
      issuer:/C=BE/O=GlobalSign nv-sa/OU=Root CA/CN=GlobalSign Root CA
      subject:/C=BE/CN=Belgium Root CA
      issuer:/C=BE/O=GlobalSign nv-sa/OU=Root CA/CN=GlobalSign Root CA
      subject:/C=BE/CN=Government CA
      issuer:/C=BE/CN=Belgium Root CA

To be considered as valid, a certificate chain received by a client should end on a root certificate that is considered as valid by the client. This implies that the client should already have the public key and thus the certificate of the root CA.

Timing cryptanalisys

- Proposed by Kocher in 1996
  - public-key crypto operations are complex and require a long time that depends on the data
  - If attacker can easily and often measure the time required to decrypt/sign some date, then it is possible to recover the private key used
- Is this applicable to SSL ?
  - Measure time between arrival of ClientKeyExchange (E(PreMasterSecret, Pub_{Bob})) and transmission of Finished message
- Countermeasures
  - add random time to each operation (not effective)
  - ensure that decryption takes fixed time

Paul Kocher's paper is available from :
http://www.cryptography.com/resources/whitepapers/TimingAttacks.pdf

Weak ciphers

- SSL supports various ciphers with various sizes of keys
  - 40 bits, 128 bits, 256 bits secret keys
  - 512, 1024, 2048 bits for RSA keys
- Client proposes ordered cipher list
  - Client should only propose strong ciphers
  - For interoperability reasons, several ciphers should be proposed by the client
- Server selects the cipher to be used
  - Server should only consider strong ciphers
  - Server should refuse sessions with weak ciphers

The following ciphers are implemented in OpenSSL (see man ciphers):

TLS v 1.0

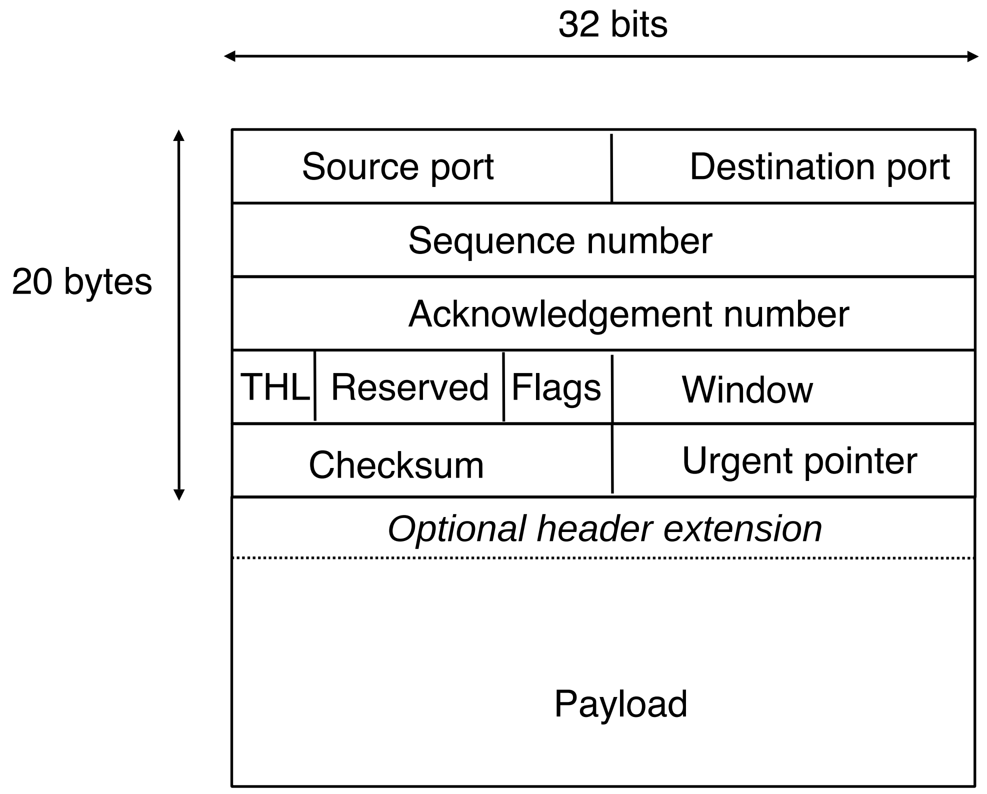
```
 TLS_RSA_WITH_NULL_MD5                   NULL-MD5
TLS_RSA_WITH_NULL_SHA                   NULL-SHA
TLS_RSA_EXPORT_WITH_RC4_40_MD5          EXP-RC4-MD5
TLS_RSA_WITH_RC4_128_MD5                RC4-MD5
TLS_RSA_WITH_RC4_128_SHA                RC4-SHA
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5      EXP-RC2-CBC-MD5
TLS_RSA_WITH_IDEA_CBC_SHA               IDEA-CBC-SHA
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA       EXP-DES-CBC-SHA
TLS_RSA_WITH_DES_CBC_SHA                DES-CBC-SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA           DES-CBC3-SHA


TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA   EXP-EDH-DSS-DES-CBC-SHA
TLS_DHE_DSS_WITH_DES_CBC_SHA            EDH-DSS-CBC-SHA
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA       EDH-DSS-DES-CBC3-SHA
TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA   EXP-EDH-RSA-DES-CBC-SHA
TLS_DHE_RSA_WITH_DES_CBC_SHA            EDH-RSA-DES-CBC-SHA
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA       EDH-RSA-DES-CBC3-SHA

TLS_DH_anon_EXPORT_WITH_RC4_40_MD5      EXP-ADH-RC4-MD5
TLS_DH_anon_WITH_RC4_128_MD5            ADH-RC4-MD5
TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA   EXP-ADH-DES-CBC-SHA
TLS_DH_anon_WITH_DES_CBC_SHA            ADH-DES-CBC-SHA
TLS_DH_anon_WITH_3DES_EDE_CBC_SHA       ADH-DES-CBC3-SHA
```

Internet and Network security

- Crypto building blocks
- Application-layer security
  - Secure Socket Layer
- Transport-layer security
  - Securing TCP
- Network-layer security

TCP packet format

- Convention
  - A TCP packet is called a segment
- TCP uses a single segment format



From the beginning, TCP relies on a single format for its 20 bytes long segment header. The TCP segment header contain several fields that will be briefly discussed later on. Among them, the flag field contain the following bit flags that indica the "function" of the TCP segment (note that one TCP segment can have several functions) :

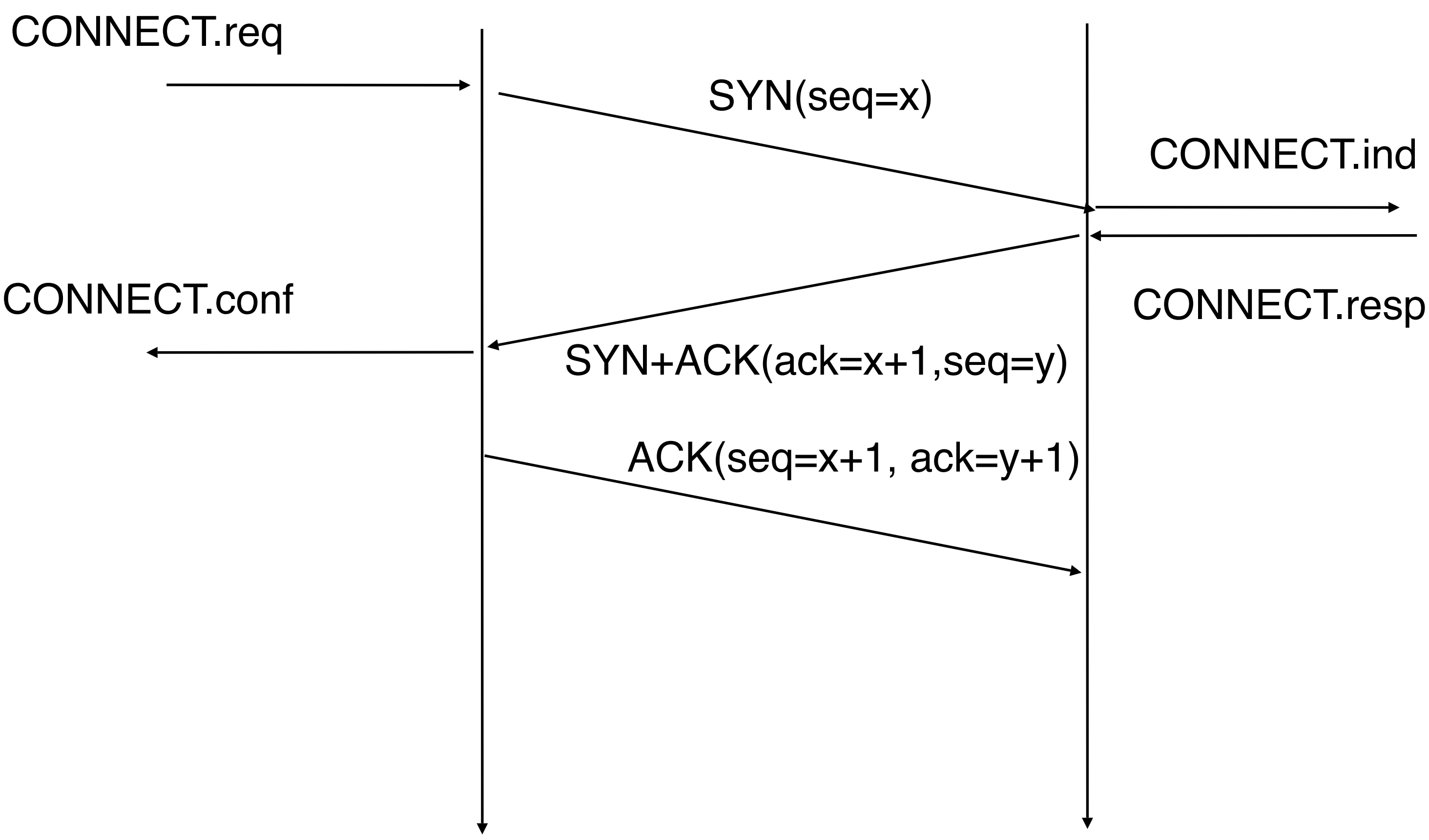
- URGent
- ACKnowledgment
- PuSH
- ReSeT
- Synchronize
- FINish

The 16 bits checksum is used to protect the payload of the TCP segment against corruption.

The optional extension header is used during connection establishment to negotiate optional features and is also used b extensions to TCP defined in [RFC1323] and [RFC2018]

Connection establishment

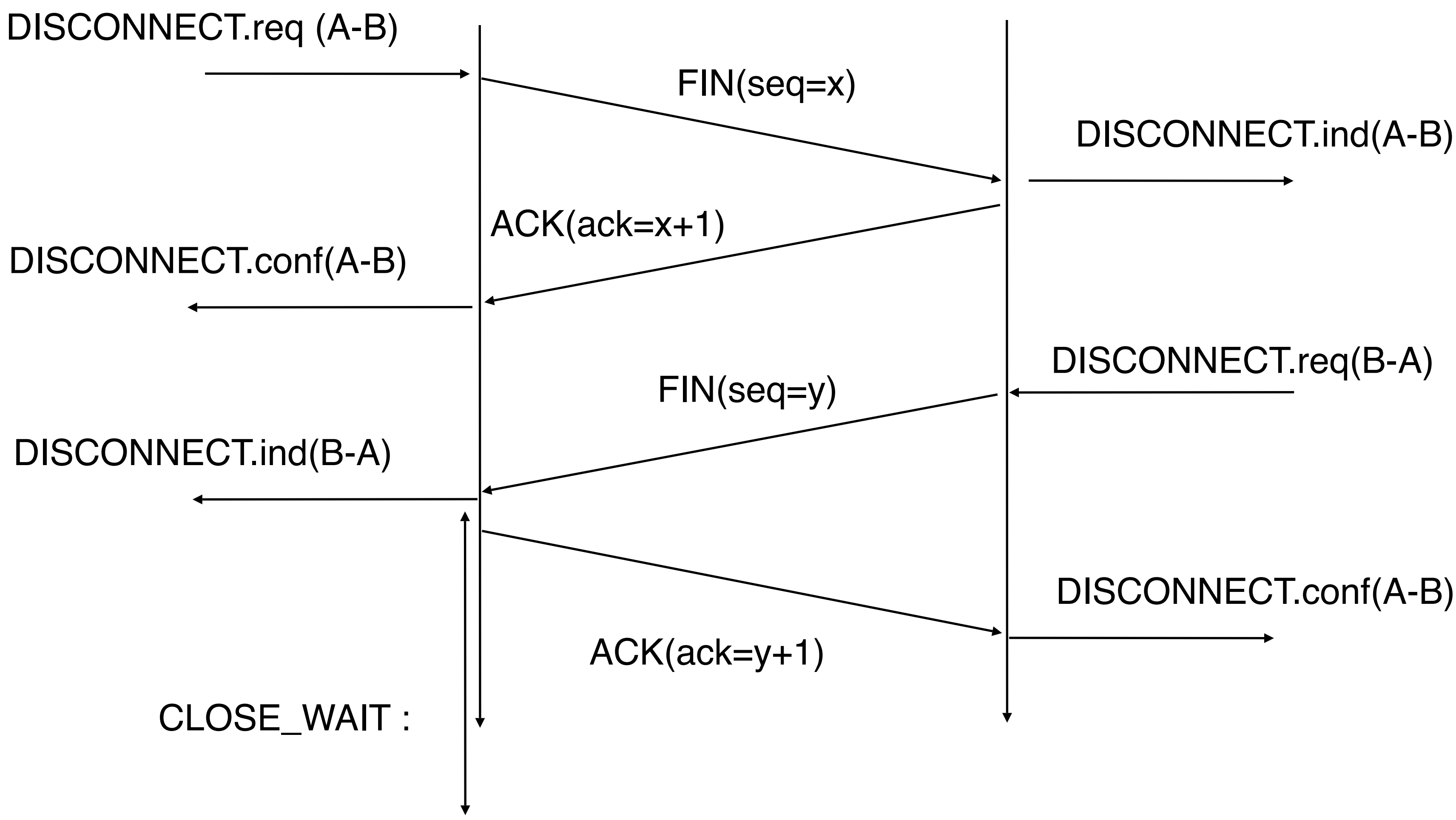
- Three-way handshake



- A TCP connection is bidirectional
  - once established, data can flow reliably in both directions

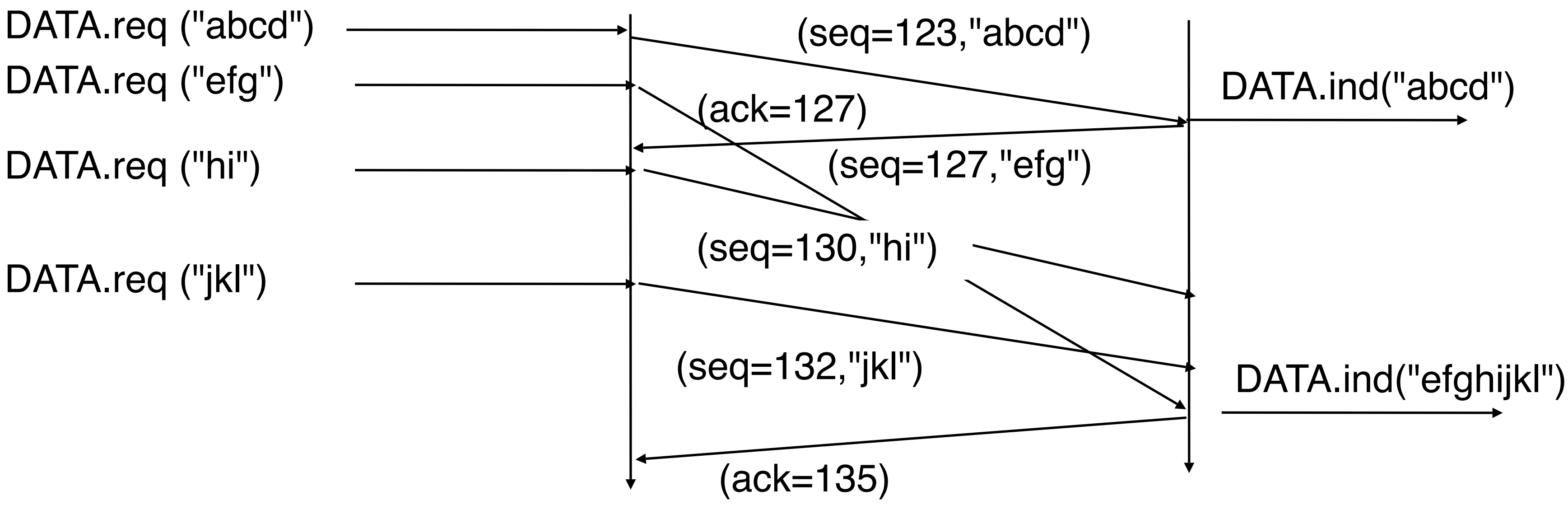
Connection release

- Independent release of the two directions



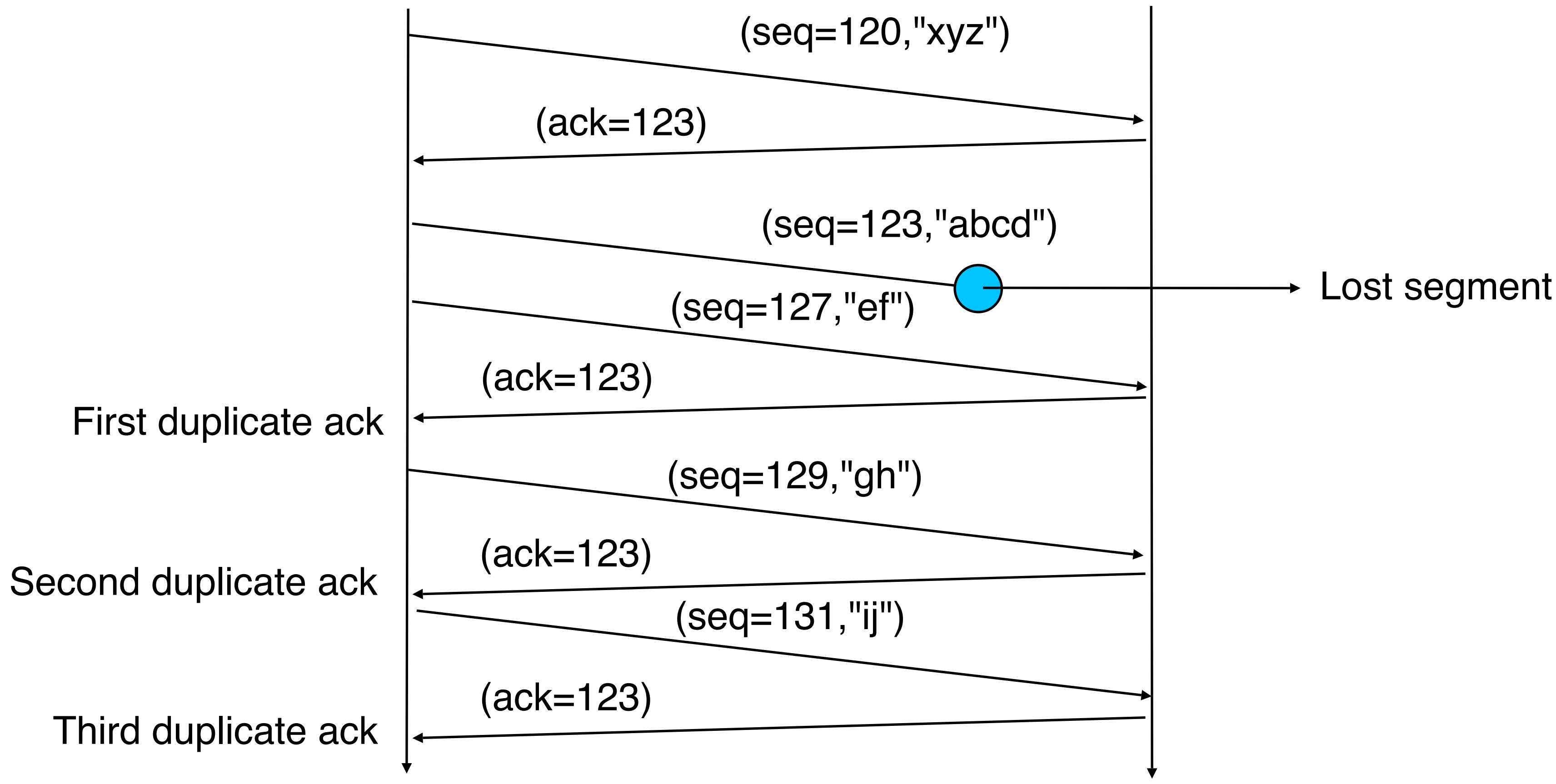
TCP : reliable data transfer

- Each TCP segment contains
  - 16 bits checksum
    - used to detect transmission errors in payload
  - sequence number (each byte consumes one number)
    - used by sender to delimit transmitted segments
    - used by receiver to reorder received segments
  - acknowledgement number
    - used by receiver (when ACK flag is set) to announce to sender the sequence number of the last byte received in sequence+1



TCP : reliable data transfer (2)

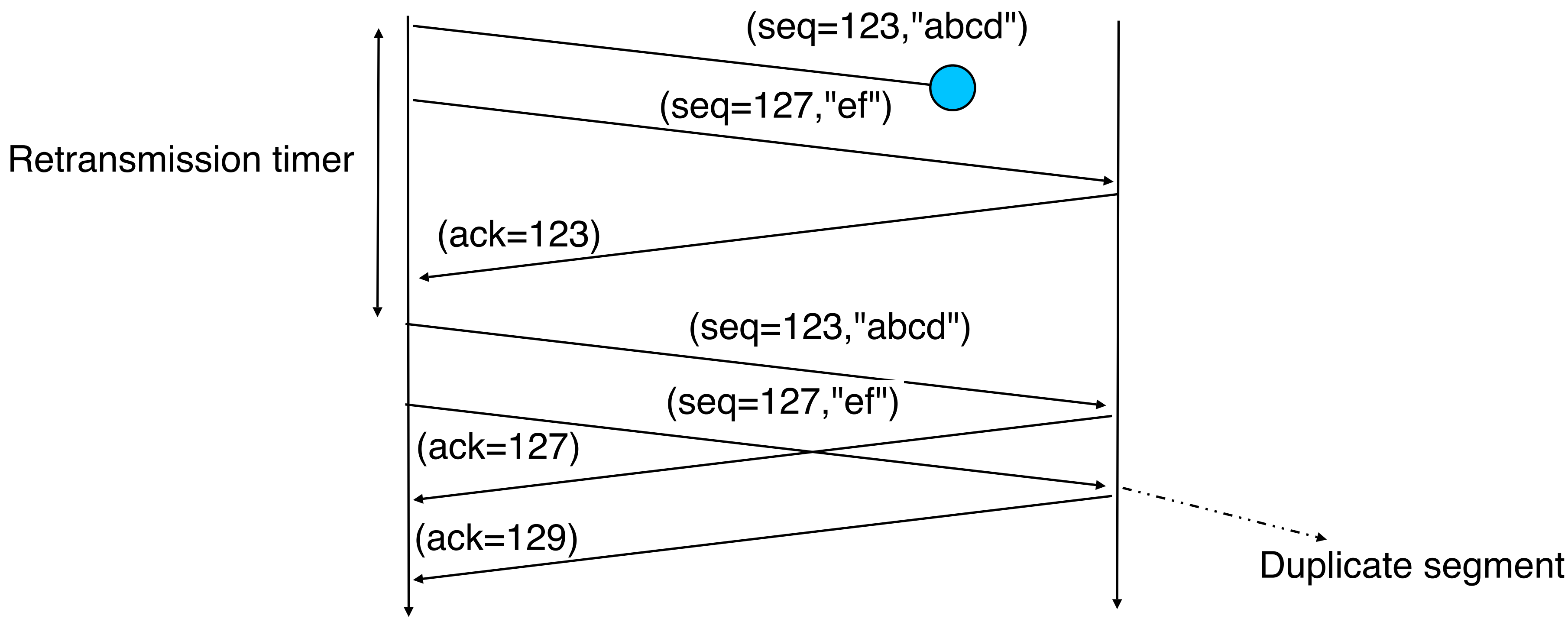
- How can we detect a lost segment ?
  - Expiration of retransmission timer
  - (three) duplicate acknowledgements



The timer based detection of the lost segments is the only mechanism that was defined in the original TCP specificati [RFC793]. All TCP implementations support it. To work properly, the TCP entity must use a reliable way to measure the roun trip-time on the TCP connection (i.e. The delay between the transmission of a TCP segment and the reception of t corresponding acknowledgment). Most TCP implementations today measure the round-trip-time as proposed in [Jacobson88]. many TCP implementations, the minimal value of the retransmission timer is around a few hundred milliseconds even if t round-trip-time is very small (such as in a LAN environment).
In addition to the default cumulative TCP acknowledgments which are supported by all TCP implementations, some TC implementations also support the Selective Acknowledgments as proposed in [RFC2018]. These SACKs are extensions to t TCP header that may be used by a receiver to inform the sender that some segments have been received out-of-sequence. the above example, the three TCP segments sent by the receiver after the loss could carry the following SACKs :
(ack=122, SACKb=[127,128]) ; (ack=122, SACKb=[127,130]),
(ack=122, SACKb=[127,132])

TCP : reliable data transfer (3)

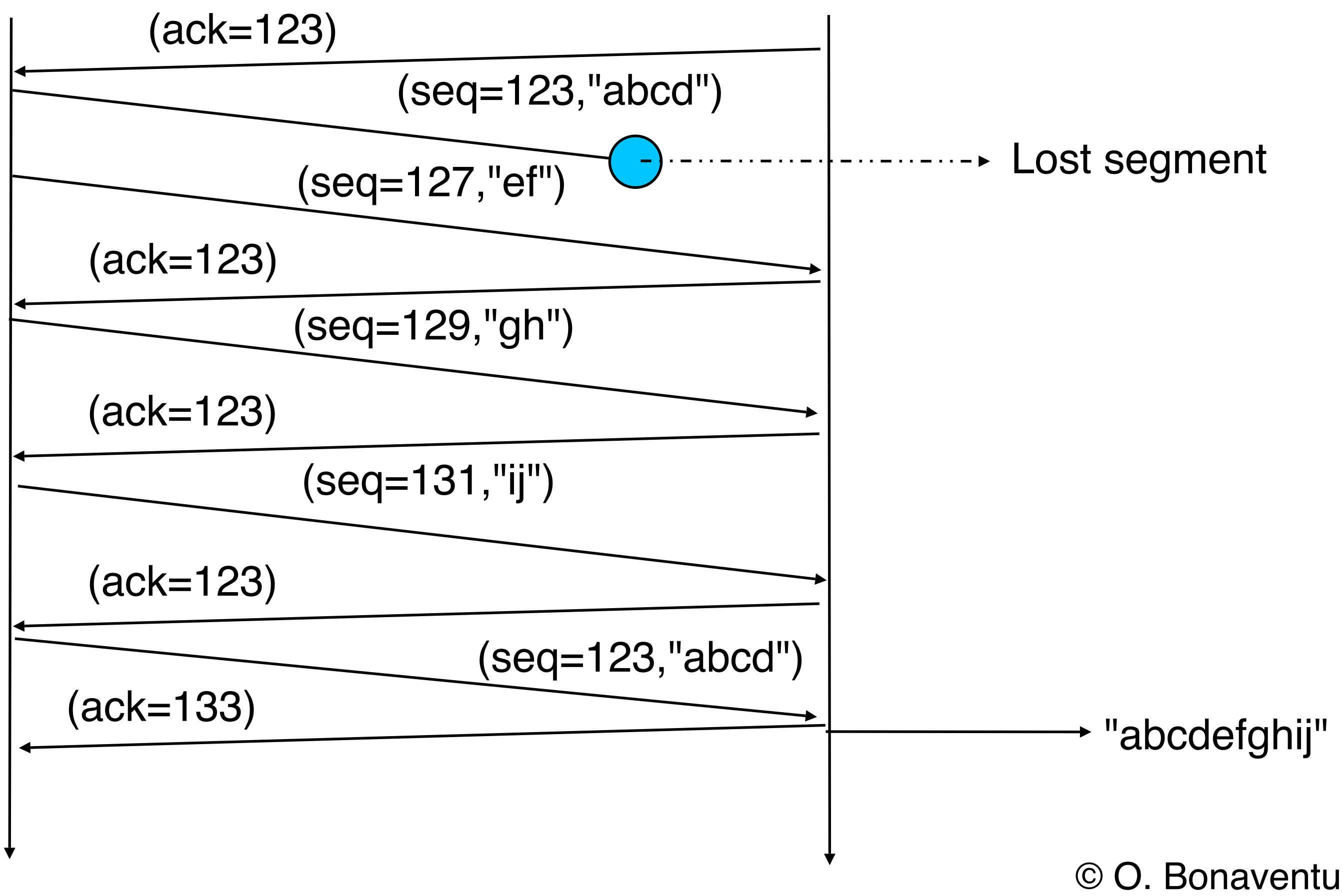
- How do we retransmit the lost segments ?
  - Upon expiration of the retransmission timer, retransmit *all* the unacknowledged segments
    - default TCP retransmission mechanism [go-back-n]



As shown above, the default TCP retransmission mechanism may retransmit segments that have already been received by th receiving TCP entity. This could be a problem on links with a high loss rate. However, in practice this retransmissio mechanism is coupled with the TCP slow-start that indirectly limits the transmission of already transmitted segments.

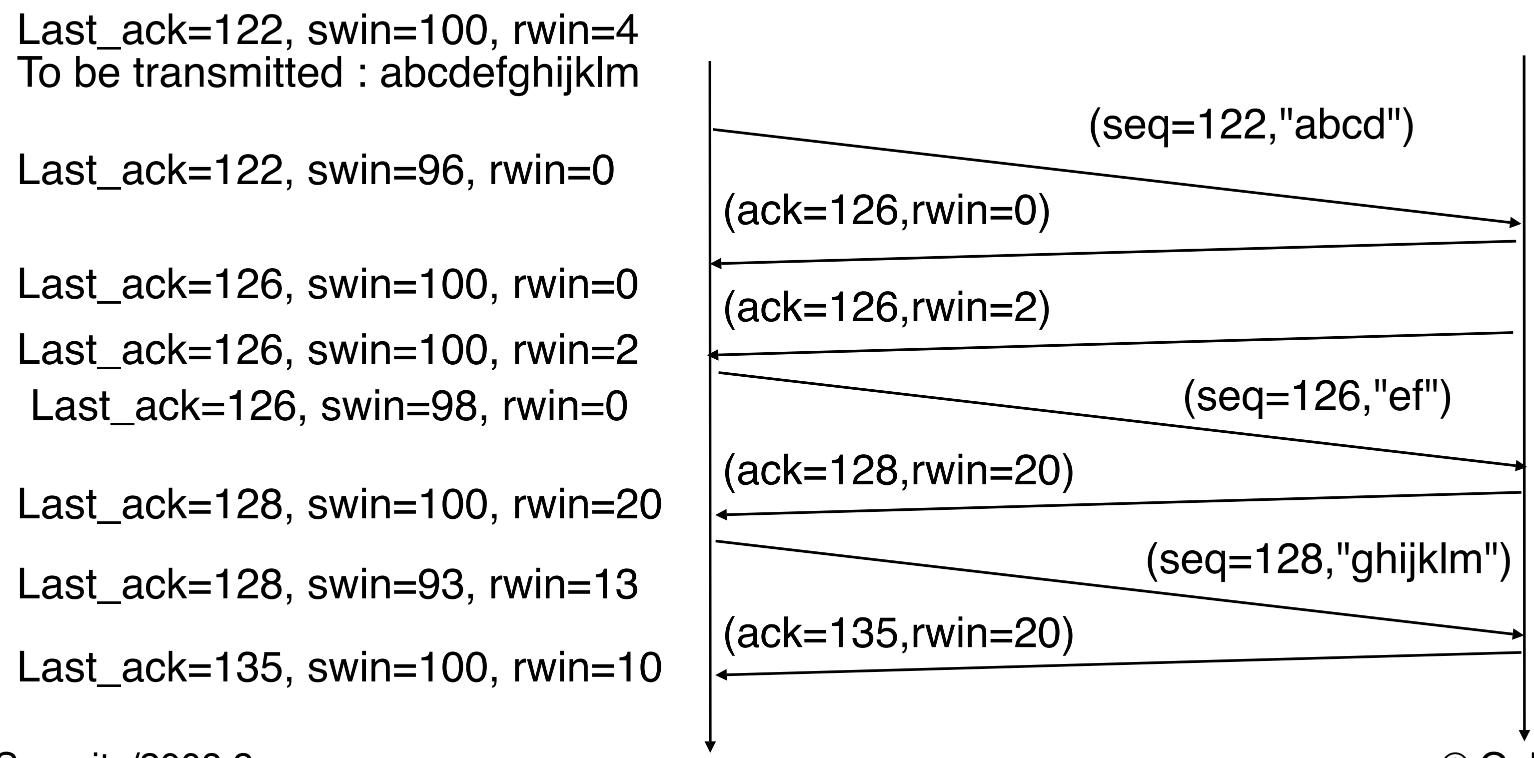
TCP reliable data transfer

- How do we retransmit the lost segments ?
  - Upon reception of three duplicate acks, retransmit *the* unacknowledged segment
    - Fast retransmit, used by most TCP implementations



TCP flow control

- Goal : protect the buffers of the receiver
- Principle
  - negotiate swin & rwin at connection establishment
  - Each TCP maintains
    - last_ack, swin, rwin



TCP flow control (2)

- Limitations
  - TCP window is encoded in a 16 bits field in the TCP segment header
    - maximum window size in normal TCP : 65535 bytes
  - Once a TCP entity has sent a complete window worth of segments, it must stop transmitting until the reception of an acknowledgement
  - Maximum throughput on a TCP connection :
    - ~ window / round-trip-time

Window \ **rtt**	**1 msec**	**10 msec**	**100 msec**
8 Kbytes	65.6 Mbps	6.5 Mbps	0.66 Mbps
64 Kbytes	524.3 Mbps	52.4 Mbps	5.2 Mbps

Window should be larger than bandwidth*delay

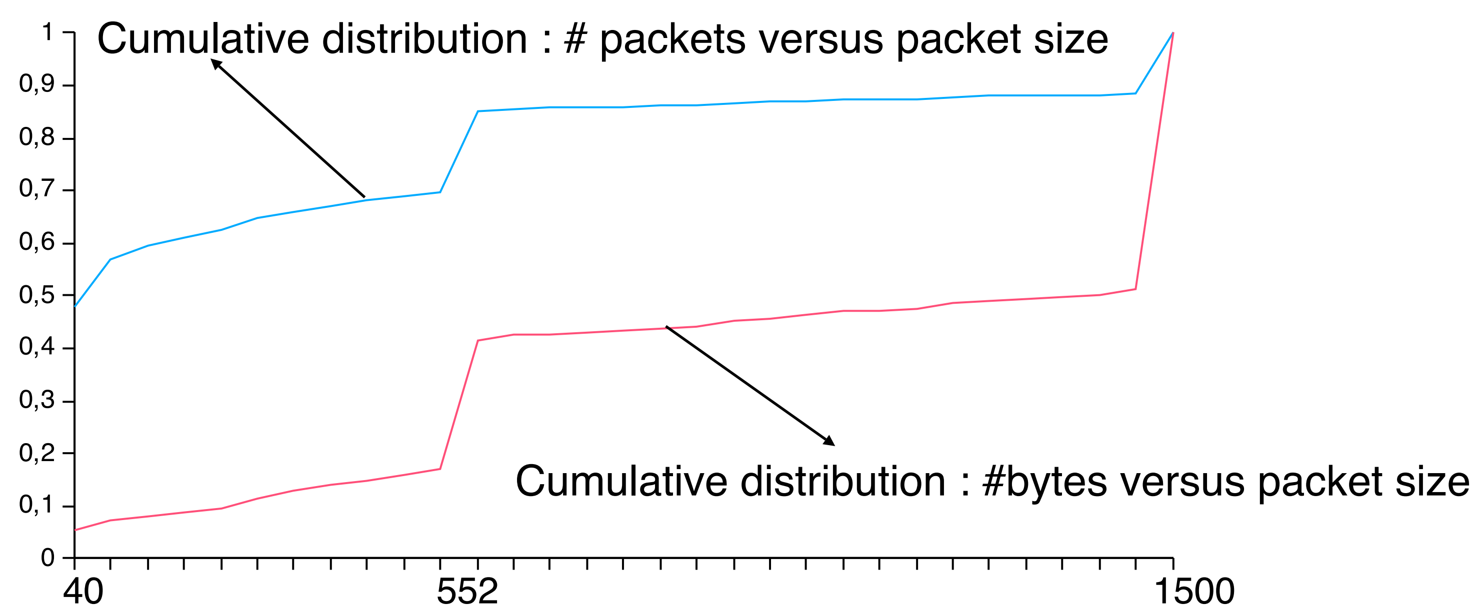
TCP segment transmission

- When do we send a TCP segment ?
  - As soon as the application gave some data to TCP
    - advantage : low delay
    - disadvantage : high overhead
  - As soon as a MSS-sized segment can be sent
    - advantage : low overhead
    - disadvantage : delay can be high

  - Nagle algorithm
    - a new segment with all the data waiting to be transmitted is sent provided that either
      - a MSS-sized segment can be sent, or
      - there is currently no segment which has already been sent but not yet acknowledged

TCP segment transmission (2)

- Consequences
  - packet size distribution is bi or multimodal
    - many ~ 40 bytes IP packets carrying only TCP acks
    - many MTU-sized IP packets carrying MSS-sized TCP segments
      - useful data is carried in large IP packets

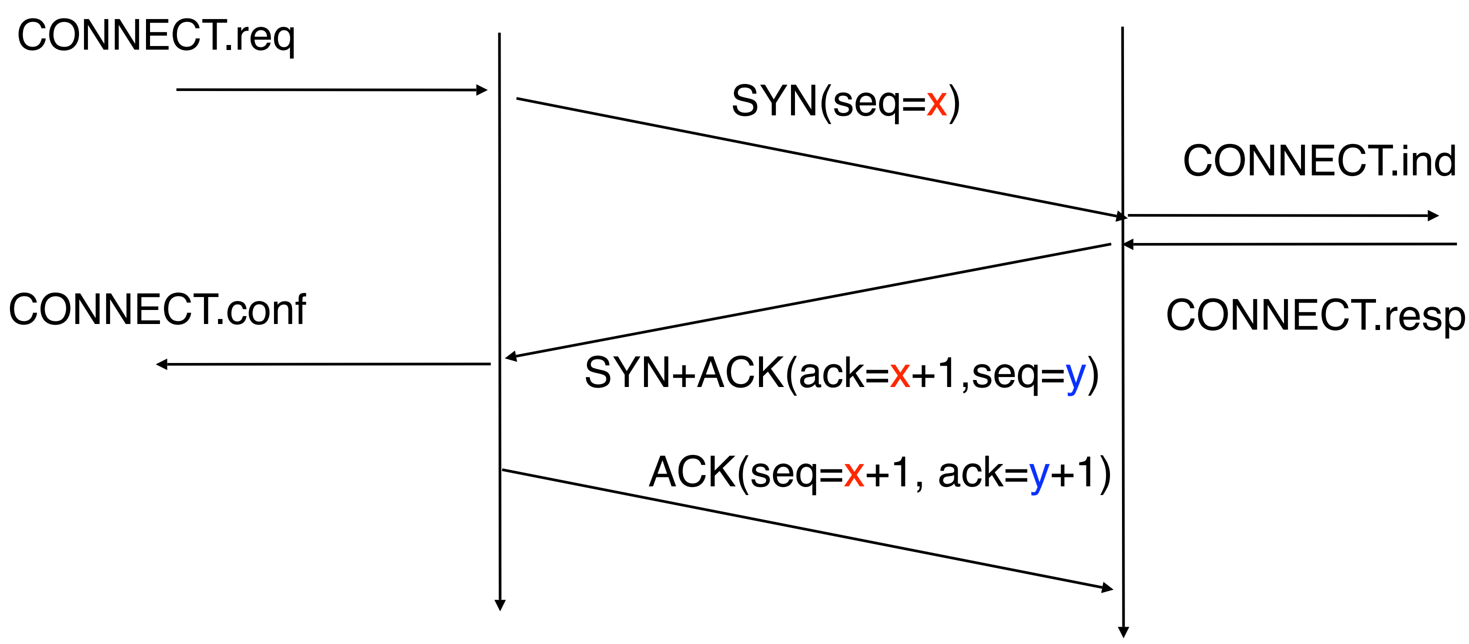


→ No such thing as an "average" size IP packet

Source : http://www.nlanr.net/NA/Learn/packetsizes.html

Packet spoofing and TCP

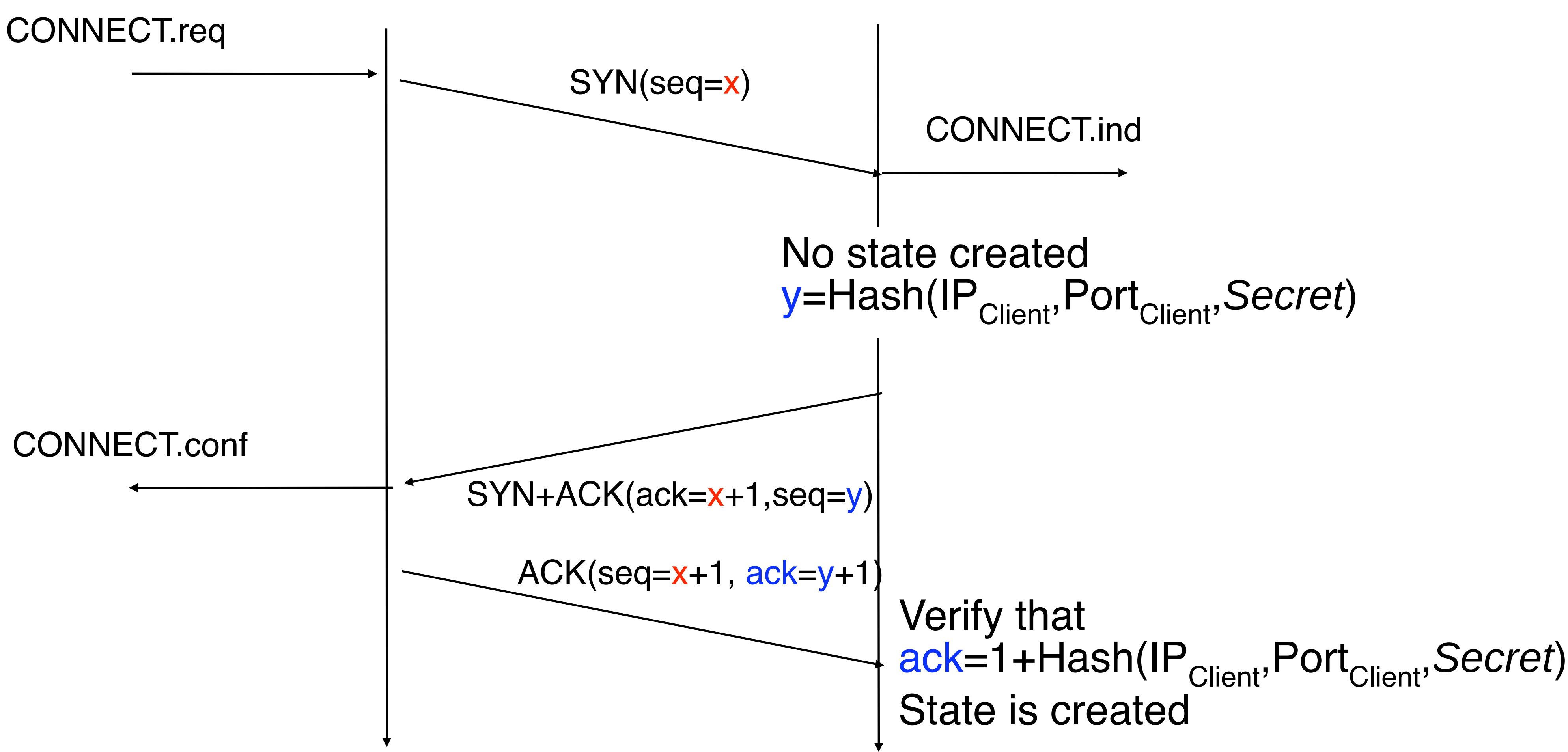
- How does packet spoofing affect TCP ?
  - In theory, three-way handshake should protect



  - In practice, things are more complex
    - Early TCP implementations read initial sequence number from counter incremented every 4 musec and after each connection establishment
      - ISS value could be predicted by an attacker

How to protect from TCP-based Denial of Service attacks ?

- Principle
  - Only store state information when the third segment of the three way handshake has been received



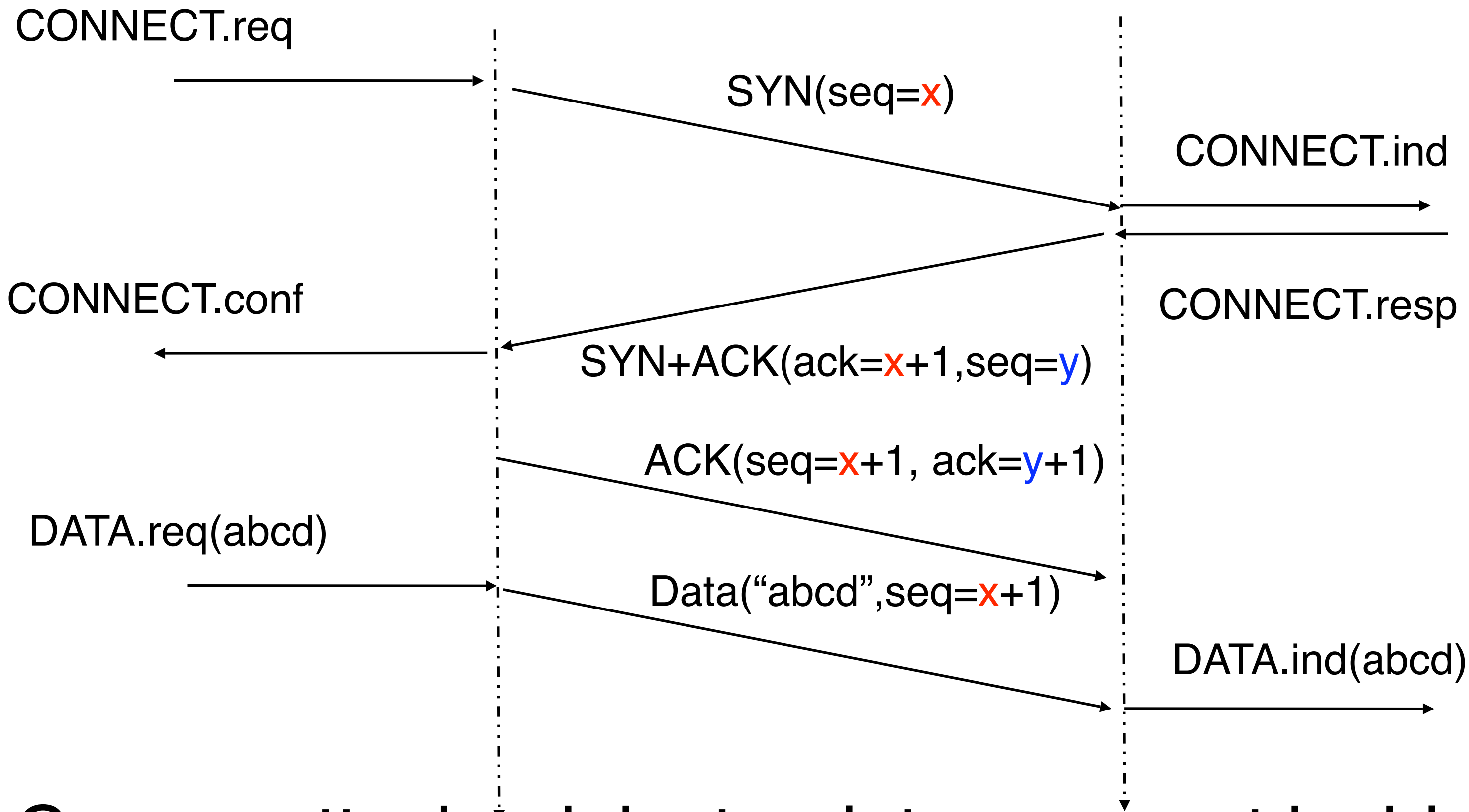
This utilization of a hash function to compute the value of the initial sequence number is usually called a SYN cookie.

In practice, the computation of the SYN cookie is slightly more complex than a simple hash function because the server must also remember inside the cookie the following information :
- the MSS value advertised by the client
- the optional utilization of TCP options such as RFC1323 large windows or timestamps or SACK by the sender

The original discussions that lead to the development of the SYN cookie solution may be found in :
http://cr.yp.to/syncookies/archive

Reliability of a TCP connection

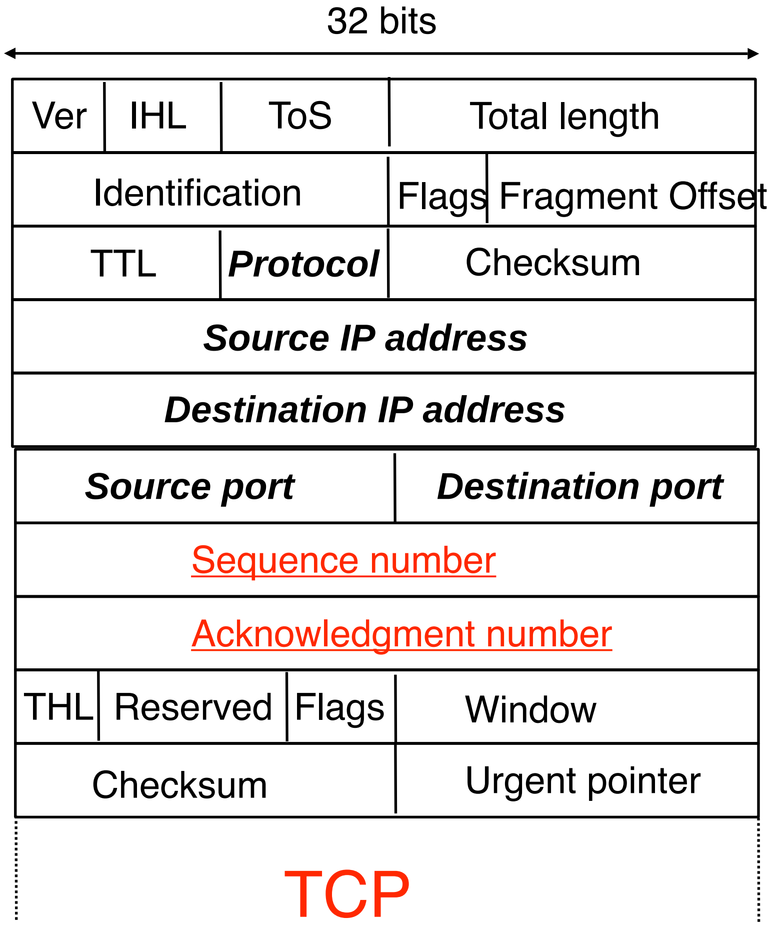
- How reliable is a TCP connection against an intelligent attacker ?



- Can an attacker inject a data segment inside an established TCP connection ?

How can an attacker inject segments in an existing TCP connection ?

- Attacker needs to build and send a TCP segment acceptable by the destination



- Attacker can capture normal segments
  - easy to inject segment if captured one
- Attacker cannot capture
  - Attacker must predict
    - Source and destination IP
    - Source and destination port
    - Easy for server, f(client OS)
    - Sequence and Ack number
    - Should be inside TCP window
    - Easier on some OSes if attacker can contact S/C

On many endsystems, the source port used by the client is simply incremented for each established TCP connection. It is thus possible to predict the TCP port number to be used. Some applications use a default port number for the client as well.

There are several ways to counter such attacks on endsystems.

The first one is to use a random initial sequence number when a TCP connection is opened. In the original TCP specification, the TCP clock was supposed to tick at a regular rate with at least one tick for each connection. With such an implementation, the initial sequence number could be easily predicted by an attacker.

One possibility to avoid such attacks is to protect the TCP connection by using MD5 hash. This solution is described in :
A. Heffernan, "Protection of BGP Sessions via the TCP MD5 Signature Option", RFC2385, August 1998.

As of today, this mechanism is mainly used to protect BGP sessions between routers.

RST attacks

- The TCP RST segment
  - sent upon reception of invalid TCP segment
    - syntax error in received segment
    - data or ack segment on invalid TCP connection
  - Reception of RST segment -> abrupt release

- Validation of received TCP RST segment
  - RST segment must contain
    - IP source and source port of active TCP connection
    - IP destination and destination port of active TCP connection
    - Sequence number of RST segment must be within received window
      - TCP sequence number space is 2^{32}, with a 64KB window, 65535 RST segments are sufficient to reset a connection

More details on this attack are available from :
M. Dalal (Ed), Transmission Control Protocol security considerations, Internet draft, draft-ietf-tcpm-tcpsecure-02.txt, November 2004
A similar attack is possible with the SYN bit instead of the RST bit.
The test for the validity of a received segment in RFC793 is :
1) If the RST bit is set and the sequence number is outside the expected window, silently drop the segment.
2) If the RST bit is set and the sequence number is acceptable i.e.: (RCV.NXT <= SEG.SEQ <= RCV.NXT+RCV.WND) then reset the connection.

Several solutions to avoid this problem are being considered, but deploying them in all TCP implementations is challenging.
A first solution is to restrict the validity check for the RST segments. A RST segment would be considered as valid only if : RCV.NXT <= SEG.SEQ <= RCV.NXT+1
With this modification, an attacker has to guess the exact sequence number.
However, this also forces the sender of a valid RST to know this information as well, which may not be possible if there are packet losses.
To avoid this problem, a possibility is to force a TCP implementation to send a ACK segment (including RCV.NXT as its ACK number) in response to the received invalid RST segment to allow the remote endsystem to respond with a RST containing the correct sequence number.

Segment injection attacks

- Issue
  - Can an attacker inject fake data segments inside an established TCP connection ?
- Information required to inject such segment
  - IP source, IP destination, src and dest ports
  - Sequence number
    - should be within the received window, typically a few tens of KBytes
  - Acknowledgement number
    - most implementations accept the received segment provided that the ack number does not ack unsent data

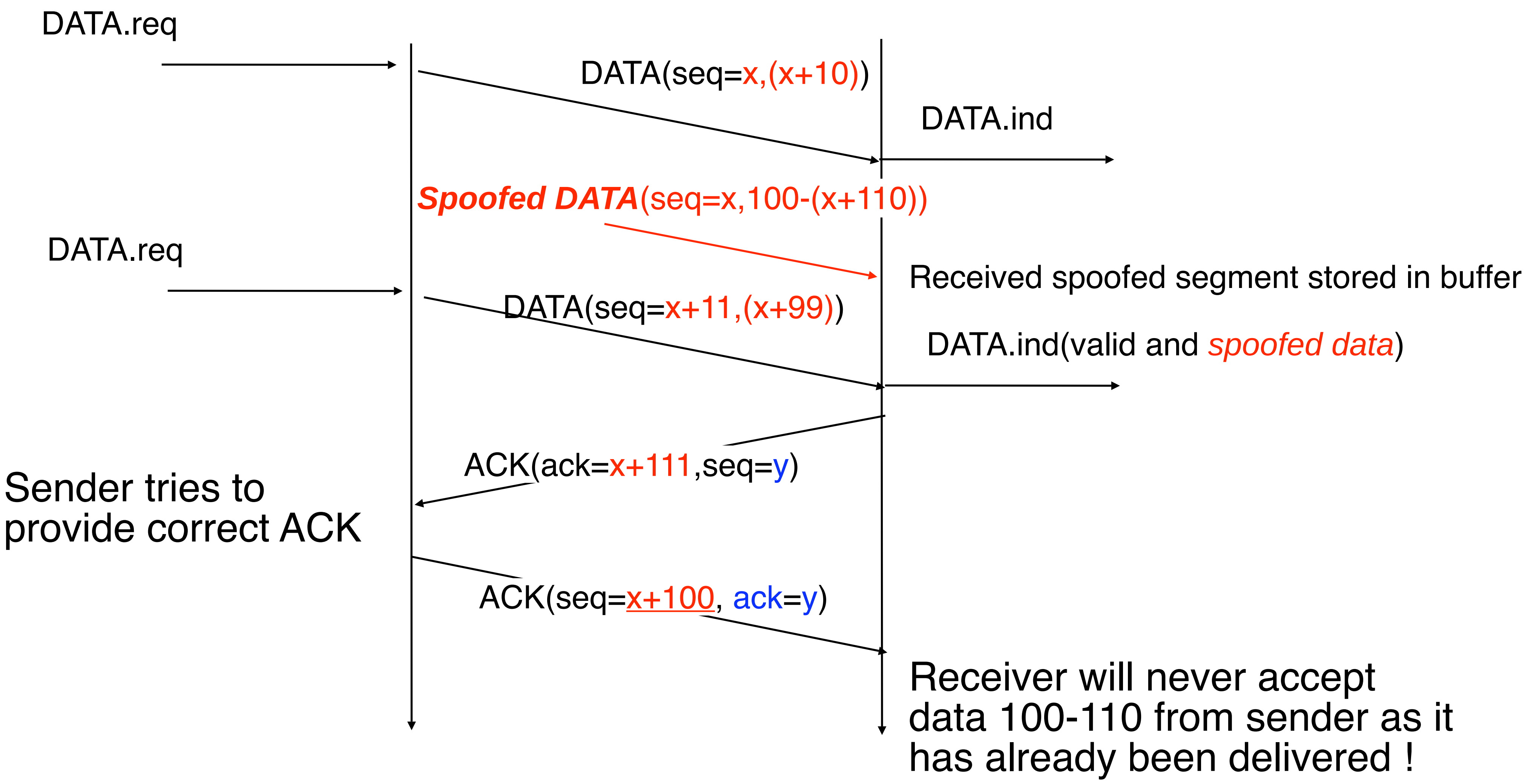
Most TCP implementations use default window sizes of a few tens of Kbytes, see http://www.psc.edu/networking/perf_tune.html

Note that implementations using much large window sizes have a higher risk as the number of spoofed data segments to be sent to find one accepted decrease hen the receiving window size increases

A possible method to reduce the risk of such attacks is to force the destination endsystems to better check the received acknowledgement number.

Segment injection attacks (2)

- What happens after an injection attack ?



- This attack can be mitigated by using two approaches :
- the first solution is to restrict the acceptable data segments by checking also the acknowledgement number when a segment is verified and rejecting the segment if the following condition is not met : (SND.UNA - MAX.SND.WND) <= SEG.ACK <= SND.NXT) (where MAX.SND.WND is the maximum value of the window ever advertised by the receiver.
- the second solution is to protect the segments sent on the TCP segments by using the MD5 option defined in RFC2385. However, this solution requires the two endpoints of the TCP connection to share a secret

With SSL, such a segment injection attack would probably cause the reception of an invalid record at the server and the SSL session would be released by the server.

Impact of TCP security issues

- SYN flooding
  - all implementations use SYN cookies to mitigate them
- Segment injection attacks
  - To succeed, attacker must send many spoofed packets and predict IP and TCP information
  - Long-lived TCP connections face higher risk than short-lived TCP connections
    - easier to spoof continuous BGP or ssh session than http
  - To reduce the impact of such attacks
    - Client should use random port numbers as often as possible among the entire port range
    - windows should not be too large

TCP MD5 option

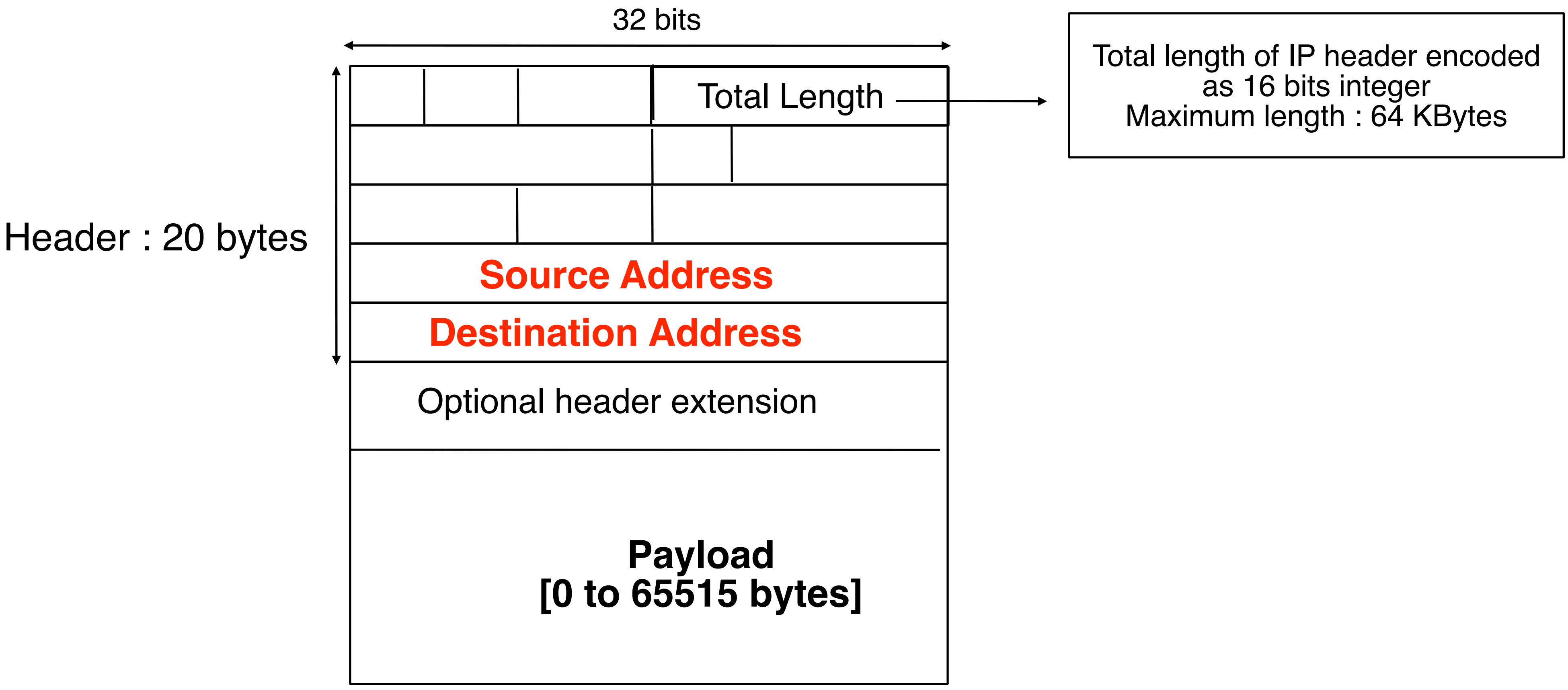
- Principle
  - TCP MD5 option negotiated during TCP connection establishment
    - MD5 option used to carry MD5 hash in each segment
  - Two endpoints of TCP connection share *secret*
  - On transmission, compute and place in segment
    - Hash = MD5 (IP source ll IP destination ll protocol number ll segment length llTCP header without options and checksum ll TCP data ll *secret*)
  - On segment arrival, recompute Hash and check
    - If MD5 option is correct, segment is processed
    - If MD5 option is incorrect, segment is discarded

Internet and Network security

- Crypto building blocks
- Application-layer security
  - SSL
- Transport-layer security
  - Securing TCP
- Network-layer security
  - IPv4
  - IPv6
  - IPSec
  - Routing security

IP Packets

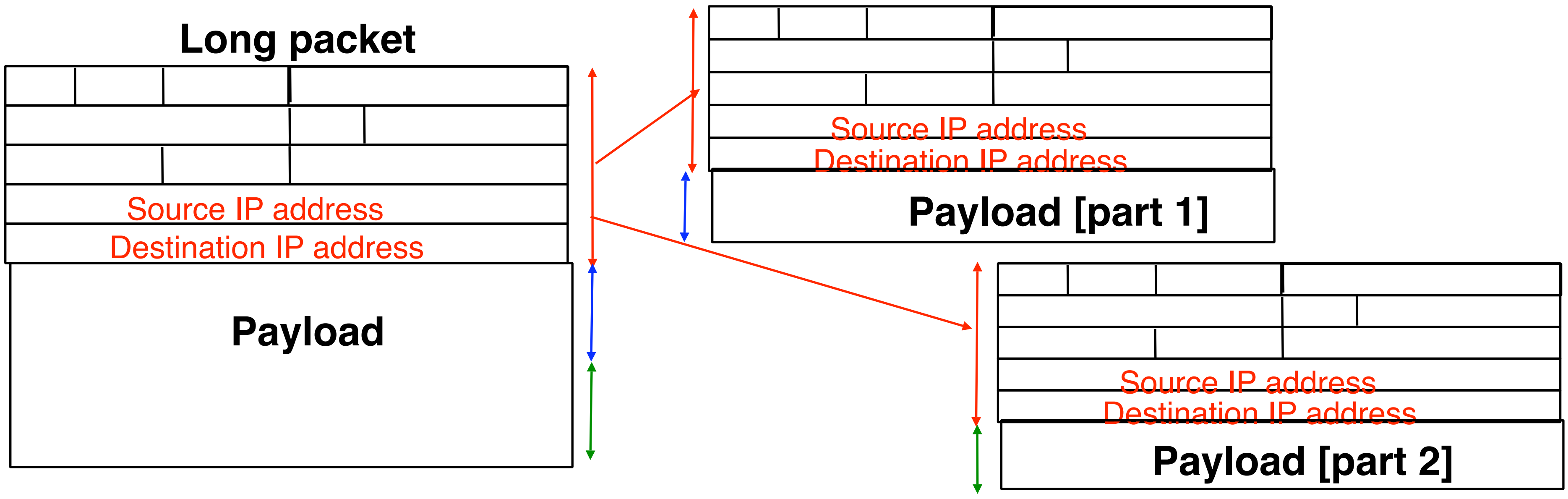
- IP packet format



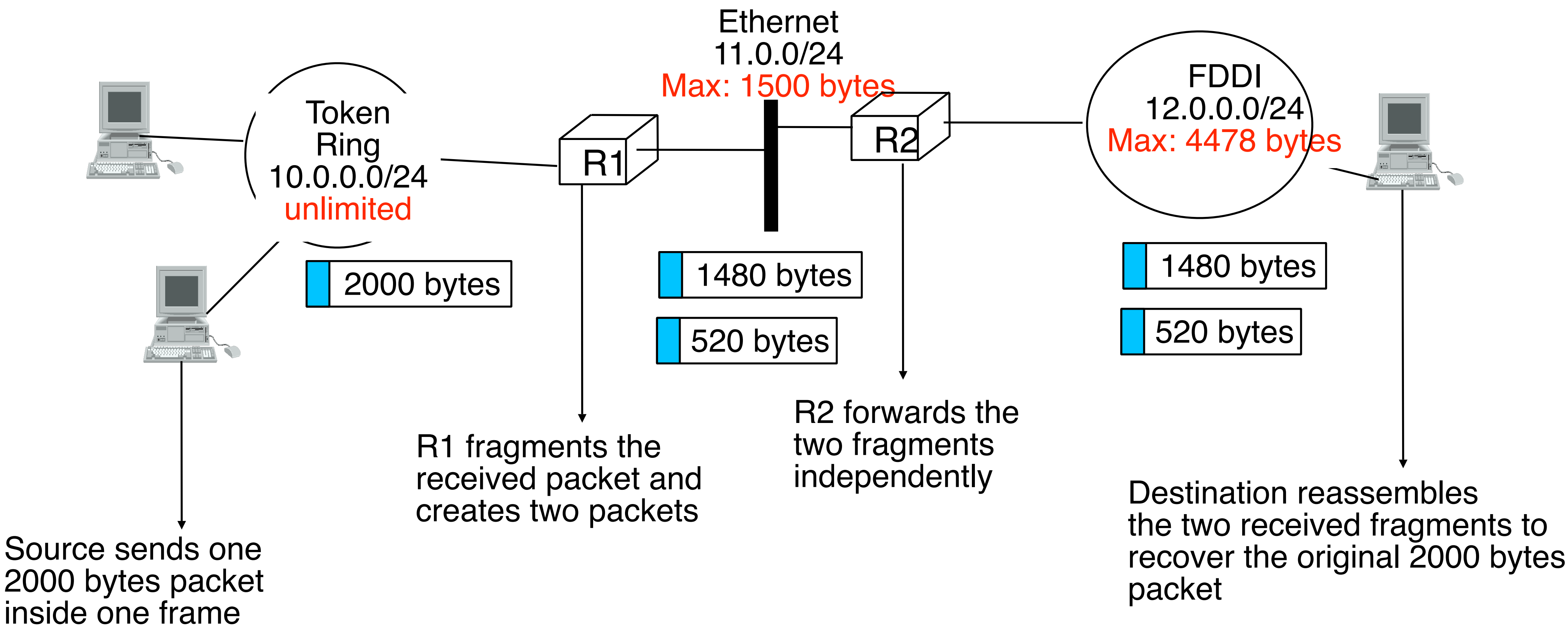
- How can we transmit a 64 KBytes packet ?

Transmission of long IP packets

- Principle
  - Each host and each router can fragment packets
    - Each fragment is a complete IP packet that contains source and destination IP addresses
  - Only the destination host performs reassembly

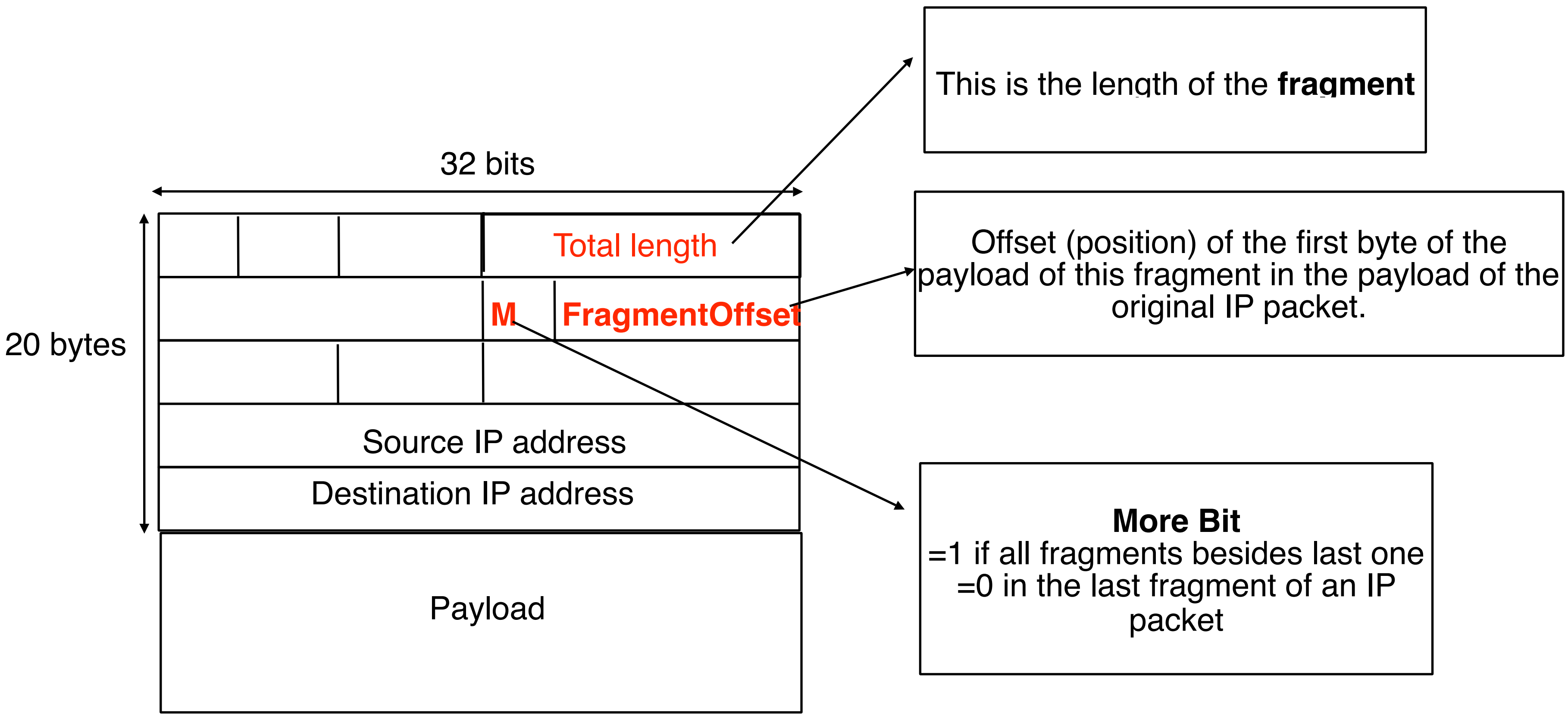


Transmission of long IP packets (2)

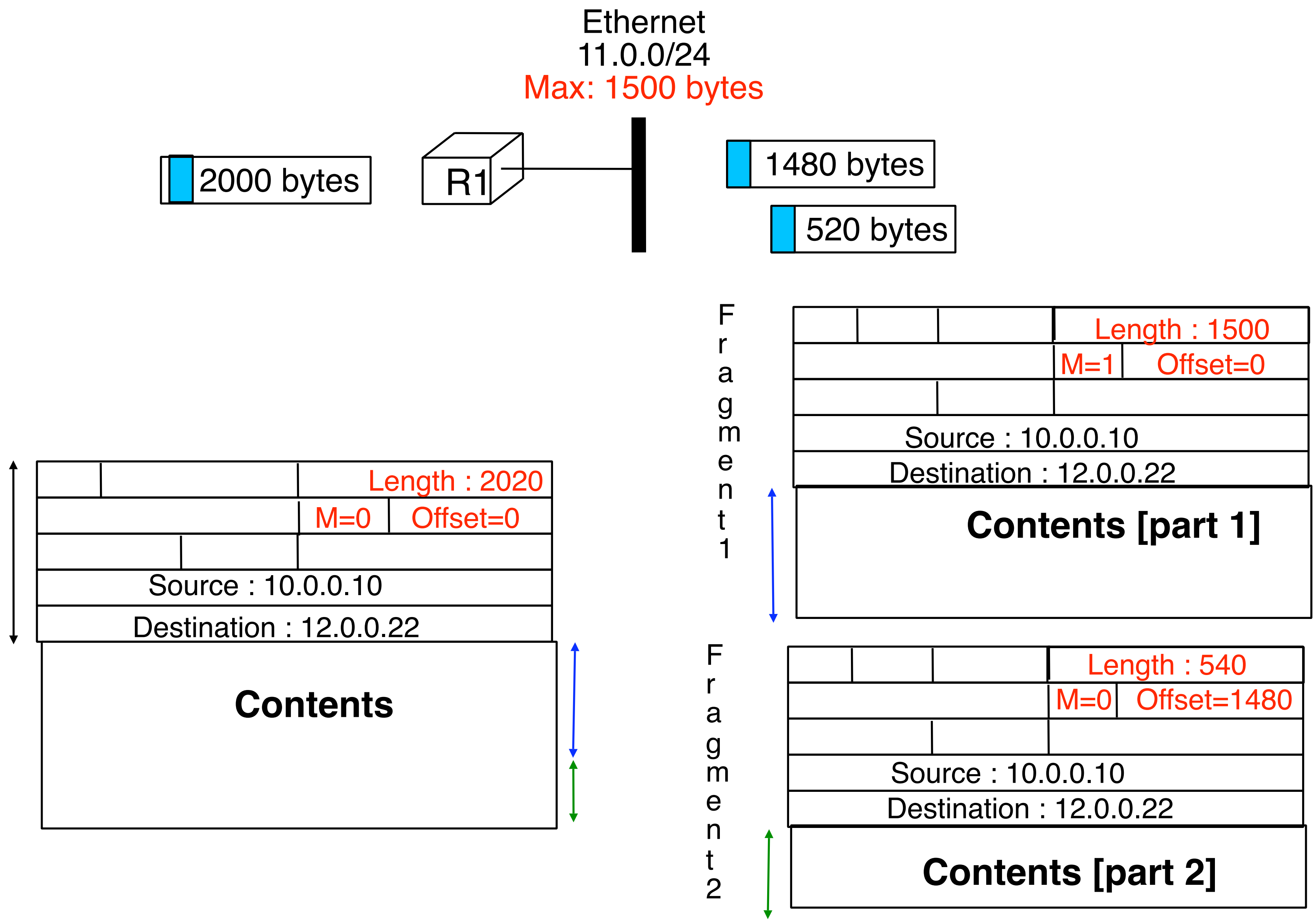


How to deal with limited MTU links ?

- IP fragmentation
  - Fragment the payload of IP packet
  - Each fragment must be numbered to recover from misordering



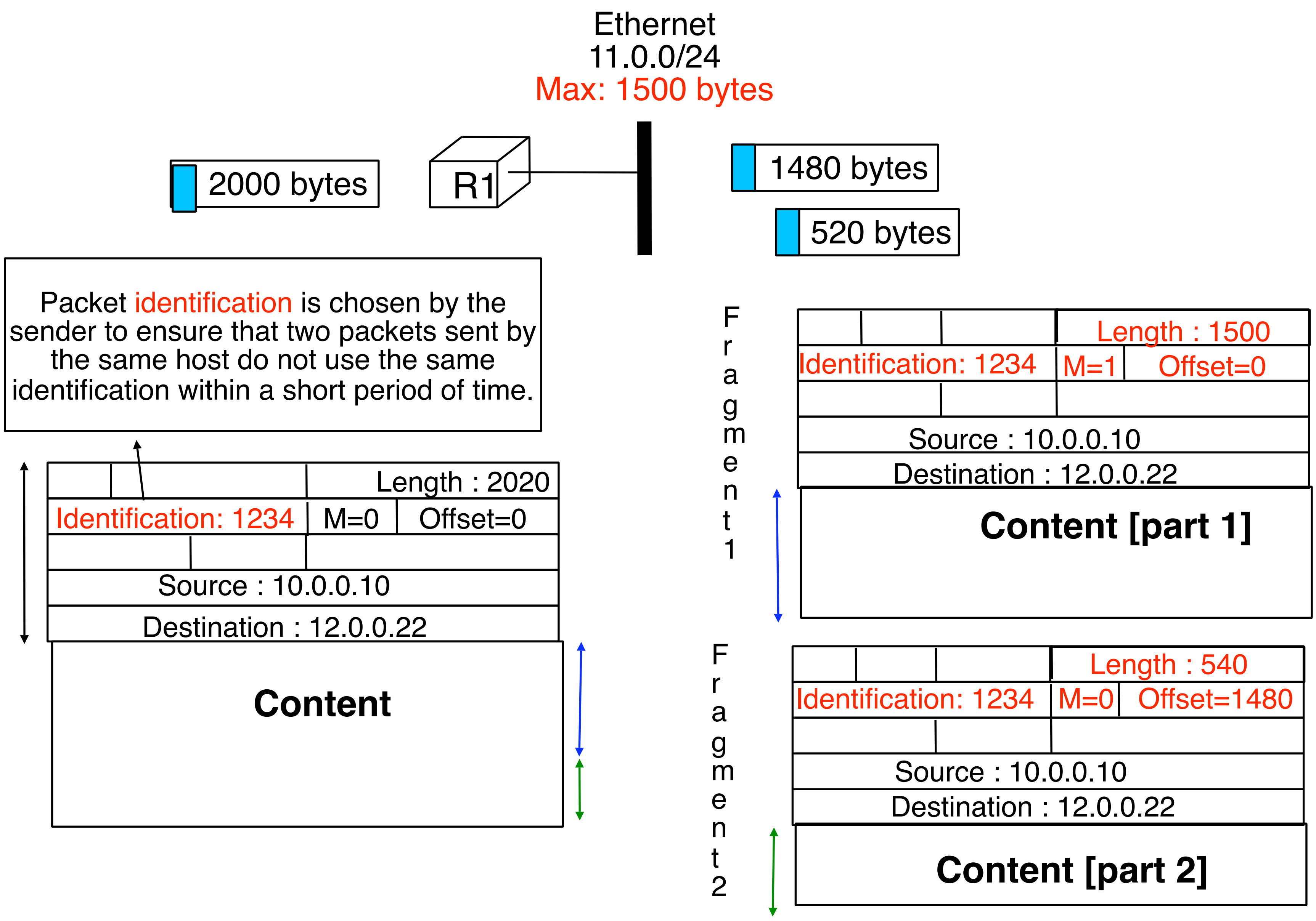
Fragmentation : example



Reassembly

- Issues
  - When does the destination has received all fragments ?
    - Last fragment contains bit More=0
    - How to handle lost fragments ?
      - the IP packet will not be reassembled by destination and received fragments of this packet will be discarded
  - How to deal with misordering
    - Offset field allows to reorder fragments from same packet
    - But misordering can cause fragments from multiple packets to be mixed
      - Each fragment must contain an identification of the original packet from which is was created

Packets and fragments identification



IP reassembly

- Basics of reassembly algorithm
  - Arrival of first fragment from packet
    - If reassembly memory is not full
      - Create data structure describing the packet
        - Some implementations allocate memory for the entire packet
      - Set reassembly timer
        - upon expiration, all fragments of this packet are dropped
    - Otherwise
      - Drop received fragment, sometimes with ICMP time exceeded

To protect the reassembly memory, implementations will usually drop new fragments earlier than fragments from partially reassembled packets when the memory becomes full. This can be implemented by using thresholds.

The reassembly memory is a limited resource on most operating systems. For example, according to Kaufman et al., Solaris allows one megabyte of reassembly memory per interface while NetBSD keeps at most 200 packets. On Solaris, the partially reassembled packets are stored during 60 seconds while they remain during 30 seconds in NetBSD. In both cases, when the reassembly buffer is full, both OS drop the incoming fragments. Thus, on NetBSD, 200 small fragments are sufficient to block the reassembly buffer for 30 seconds, while for Solaris, one MB of fragments is required for 60 seconds. This creates a risk of DoS against application-layer protocols that rely on IP fragments, such as the applications transmitting large SDUs over UDP.
C. Kaufman, R.Perlman, B. Sommerfeld, DoS protection for UDP-based protocols, CCS03, October 2003, Washington, USA

The ping of death was an attack against the reassembly algorithm on machines using some variants of the Windows operating system. On such machines, it was possible to cause the OS to crash by sending a specially crafted packet containing more than 65535 bytes. This OS was not prepared to handle such fragments and this cause a buffer overflow problem inside the OS.

IP reassembly (2)

- Arrival of next fragment from packet
  - If reassembly memory is not full
    - Add fragment to data structure corresponding to packet
  - Otherwise
    - Discard fragment and partially reassembled packet

- Security issues
  - Reassembly memory is often limited -> DoS risk
    - A source may block IP fragment reassembly at a destination by sending too many small fragments
  - ping of death
    - Some operating systems had difficulties when receiving packets containing more than 64 KBytes and in some cases crashed

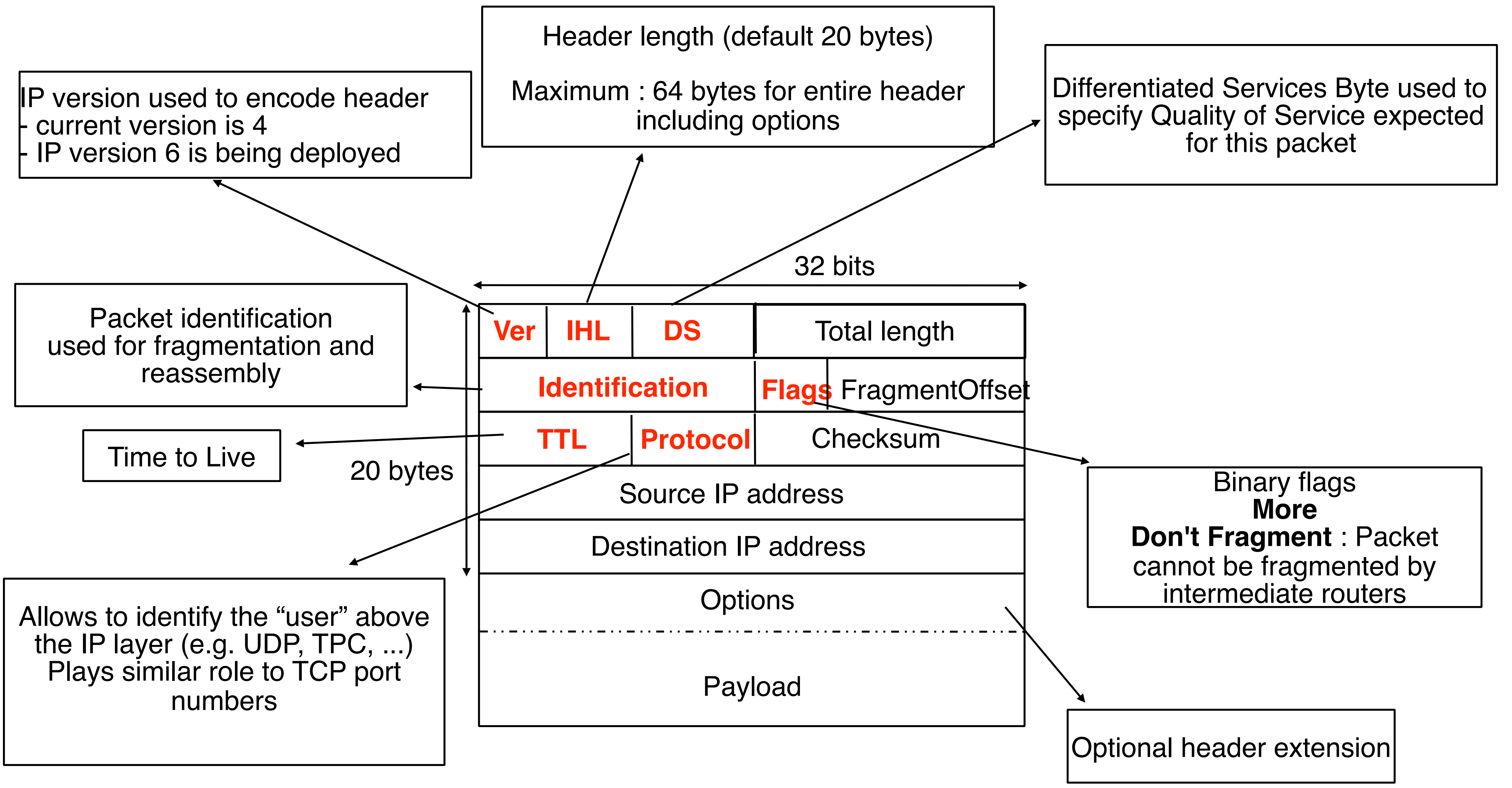
Transmission errors

- How should IP react to transmission errors ?
  - Transmission error inside packet content
    - some applications may continue to work despite this error
    - IP : no detection of transmission errors in packet payload
  - Transmission error inside packet header
    - could cause more problems
      - imagine that the transmission error changes the source or destination IP address
    - IP uses a checksum to detect transmission errors in header
      - 16 bits checksum (same as TCP/UDP) computed only on header
      - each router and each end host verifies the checksum of all packets that it receives. A packet with an errored header is immediately discarded

Transient and permanent loops

- Problem
  - Loops can occur in an IP network
    - permanent loops due to configuration errors
    - transient loops while routing tables are being updated
- Solution
  - Each packet contains a Time-to-Live (TTL) that indicates the maximum number of intermediate routers that the packet can cross
    - many hosts set the initial TTL of their packets to 32 or 64
  - each router checks the TTL of all packets
    - If TTL=1, packet is discarded and source is notified
    - If TTL>1, packet is forwarded and TTL is decremented by at least 1
      - routers thus must recompute checksum of all forwarded packets
  - Utilisation of TTL is a means to bound the lifetime of packets inside the Internet

IP header format



Protocol field

1	ICMP	Internet Control Message	[RFC792]
2	IGMP	Internet Group Management	[RFC1112]
4	IP	IP in IP (encapsulation)	[RFC2003]
6	TCP	Transmission Control	[RFC793]
17	UDP	User Datagram	[RFC768]

Voir http://www.iana.org/assignments/protocol-numbers

IP Options

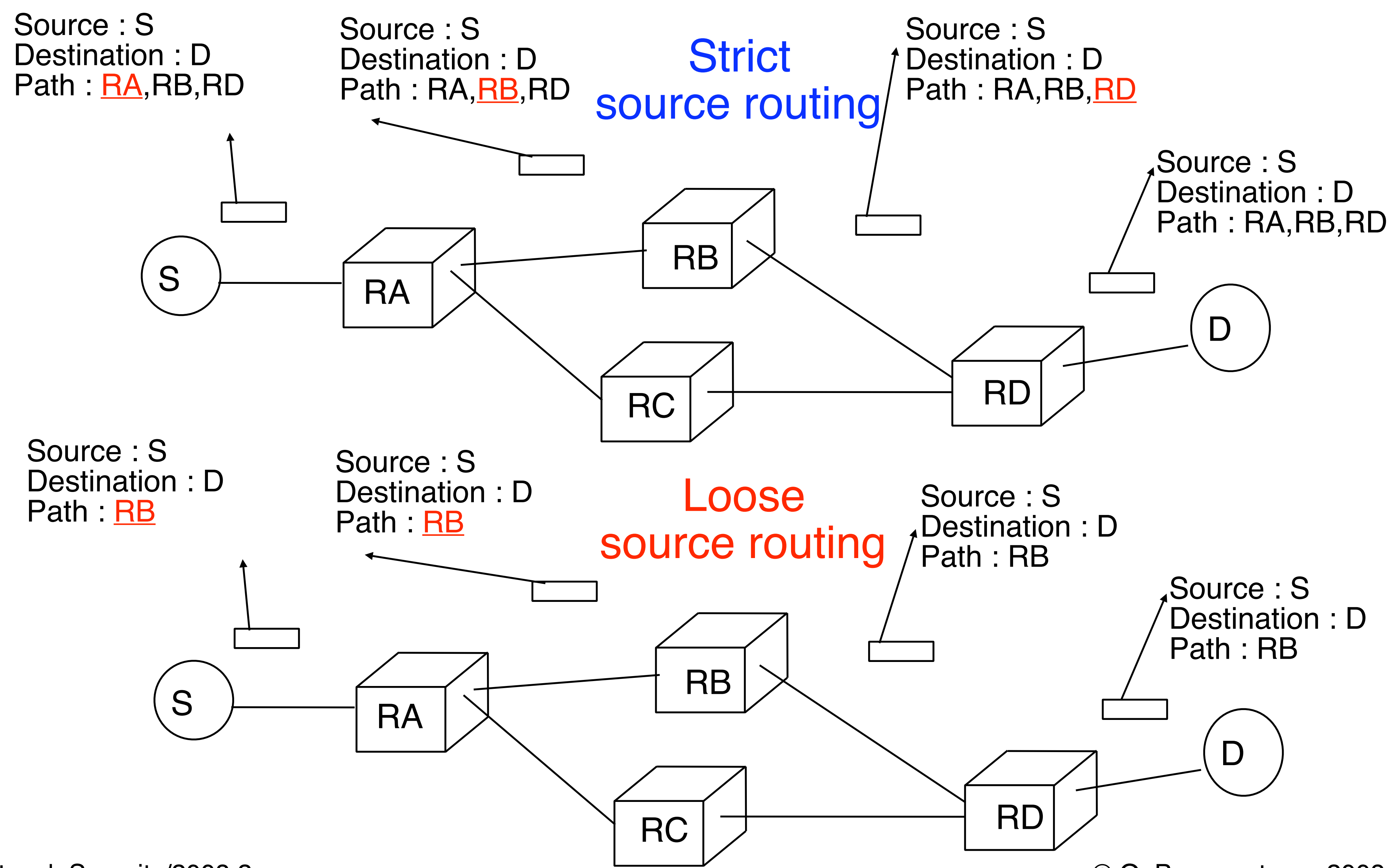
- Sample IP header options
  - Strict source route option
    - allows the source to list IP addresses of all intermediate routers to reach destination between source and destination
  - Loose source route option
    - allows the source to list IP addresses of some intermediate routers to reach destination between source and destination
  - Record route option
    - allows each router to insert its IP address in the header
      - rarely used because limited header length
  - Router alert
    - allows the source to indicate to routers that there is something special to be done when processing this packet

Constraint : maximum header size with option 64 bytes

RFC791 Internet Protocol. J. Postel. Sep-01-1981.

RFC2113 IP Router Alert Option. D. Katz. February 1997

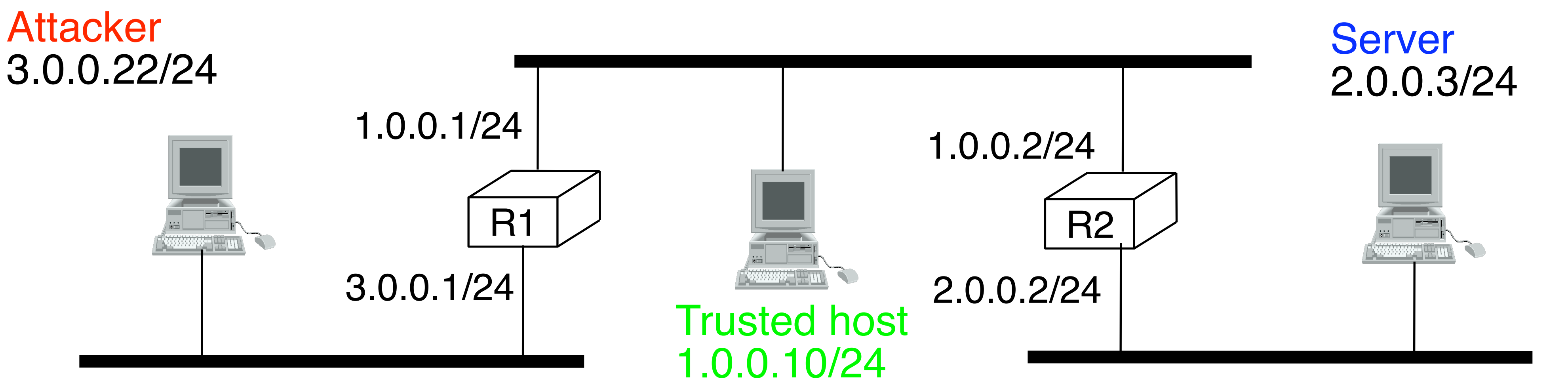
IP Source Routing



Le format de l'extension d'entête IP permettant de supporter le routage spécifié par la source East le suivant :

IP source routing

- Principle
  - Each packet contains a list of transit routers
    - Allows hosts to decide the route of their packets
    - When replying to source routed packets, hosts *reverse the source route in the received packet*
- Security risk
  - A host can easily impersonate another one



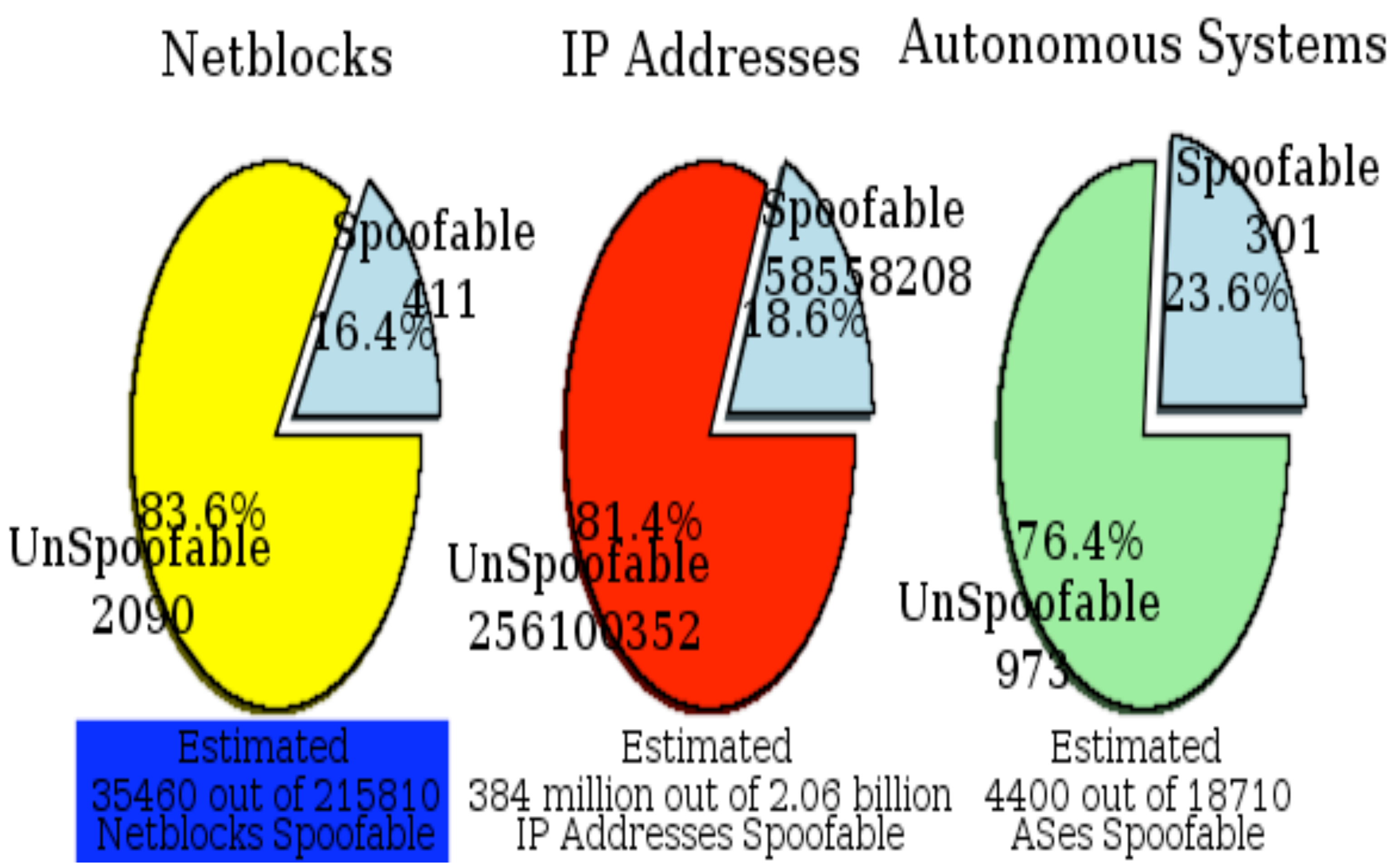
- In the example above, the attacker can send spoofed packets with source=10.0.0.10 and destination 2.0.0.3 and add to each packet a source route option indicating that the list of intermediate routers are :
- 3.0.0.22
- 1.0.0.1
- 2.0.0.2

Upon reception of such packets, the server will install a source route to reach IP address 1.0.0.10 via the attacker. This allows the attacker to send and receive packets as if it was using IP address 1.0.0.10.

In most networks, source routing is disabled and routers drop packets containing the source routing option. The legitimate utilizations of source routing are so rare today that this is not a problem.

IP Packet spoofing

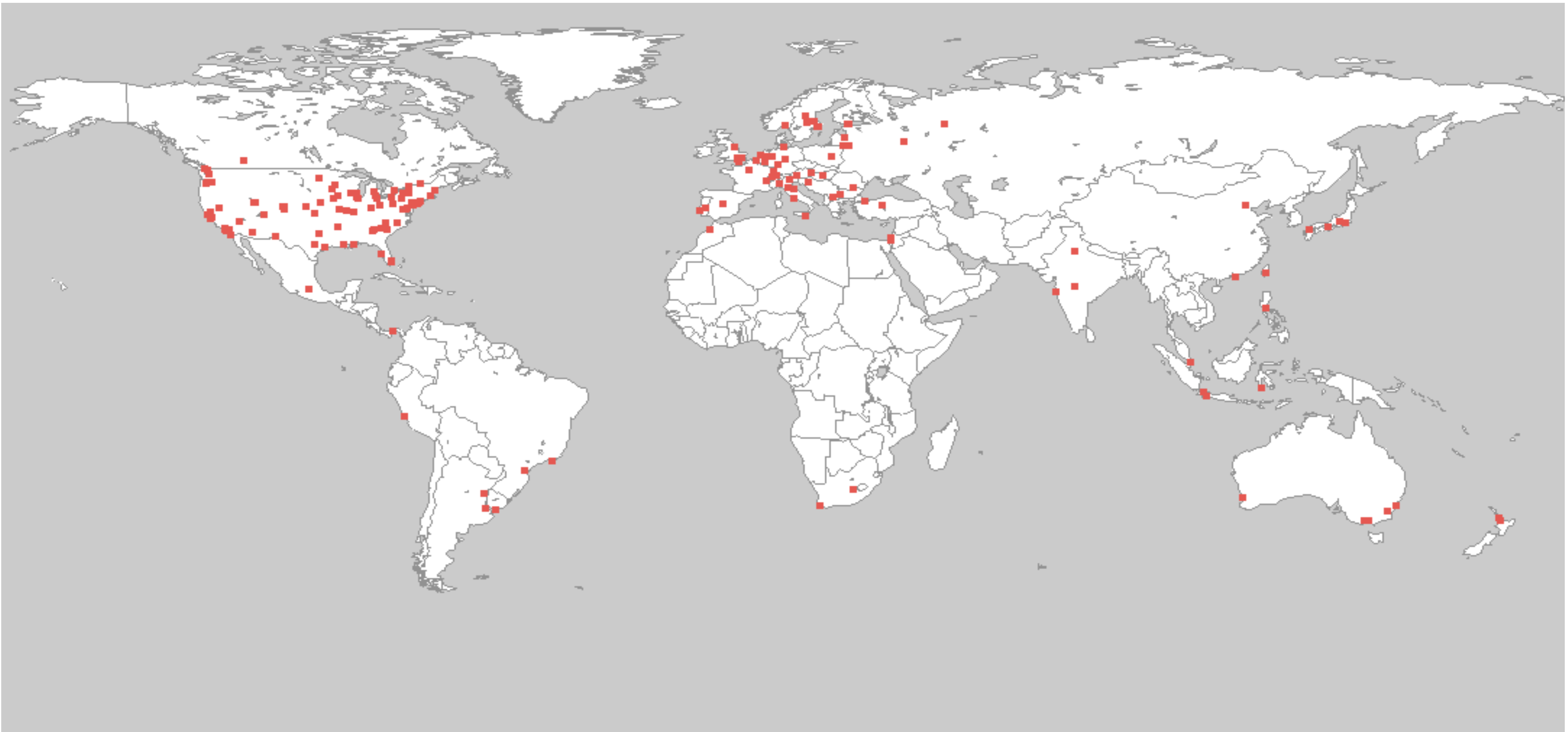
- How important is the problem ?



Based on a study perfomed at MIT : http://spoofer.csail.mit.edu/summary.php

IP Packet spoofing (2)

- How widespread is the problem ?



IP Packet spoofing (3)

- What can be done to avoid spoofing ?
  - Ingress filters
    - configure border routers of enterprise network to reject all packets whose source address belongs to the IP prefixes of the enterprise
  - RPF check
    - Principle
      - When a packet arrives from source S on *interface i*, consult routing table to check that route to S is via *i*
        - If yes, packet can be forwarded
        - Otherwise, packet is dropped
    - Limitation
      - Does not protect against spoofing from the LAN containing the subnet of the spoofed address

Ingress filtering is defined in :

P. Ferguson, D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. May 2000, RFC2827

Operation of an IP endhost

- Required information on an IP endhost
  - IP addresses of its interfaces
    - For each address, the subnet mask allows the endhost to determine the addresses that are directly reachable through the interface
  - (small) routing table
    - Directly connected subnets
      - From the subnet mask of its own IP addresses
    - Default router
      - Router used to reach any unknown address
      - By convention, default route is 0.0.0.0/0
    - Other subnets known by endhost
      - Could be manually configured or learned through routing protocols are special packets (see later)

Example

```
/sbin/ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
        inet 127.0.0.1 netmask ff000000
hme0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
        inet 130.104.229.58 netmask ffffff80 broadcast 130.104.229.127
```

Cette station dispose de deux interfaces, l'interface loopback East lo0 et l'interface Ethernet hme0.

table de routage

netstat -rnv

```
IRE Table:
  Destination          Mask          Gateway          Device Mxfrg  Rtt  Ref Flg  Out  In/Fwd
-------------------- --------------- -------------------- ------ ----- ----- --- --- ----- ------
130.104.229.0        255.255.255.128 130.104.229.58       hme0    1500*    0   3 U     5750     0
224.0.0.0            240.0.0.0       130.104.229.58       hme0    1500*    0   3 U        0     0
default              0.0.0.0         130.104.229.126              1500*    0   0 UG   42564     0
127.0.0.1            255.255.255.255 127.0.0.1            lo0     8232*  315   0 UH   65966     0
```

default correspond à la route par défaut, 0.0.0.0/0 et 224.0.0.0 correspond au multicast

IP address configuration

- How does a host know its IP address
  - Manual configuration
    - Used in many small networks
  - Server-based autoconfiguration RARP
    - DHCP
      - Dynamic Host Configuration Protocol
      - Principle
      - When it attaches to a subnet, endhost broadcasts a request to find DHCP server
      - DHCP server replies and endhost can contact it to obtain IP address
      - DHCP server allocates an IP address for some time period and can also provide additional information (subnet, default router, DNS resolver, ...)
        - DHCP servers can be configured to always provide the same IP address to a given endhost or not
        - Endhost reconfirms its allocation regularly

RFC2131 Dynamic Host Configuration Protocol. R. Droms. March 1997.

Operation of an IP router

- Required information on an IP router
  - IP addresses of its interfaces
    - For each address, the subnet mask allows the endhost to determine the addresses that are directly reachable through the interface
  - Routing table
    - Directly connected subnets
      - From the subnet mask of its own IP addresses
    - Other known subnets
      - Usually learned via routing protocols, sometimes manually configured
    - Default router
      - Router used to reach any unknown address
      - By convention, default route is 0.0.0.0/0

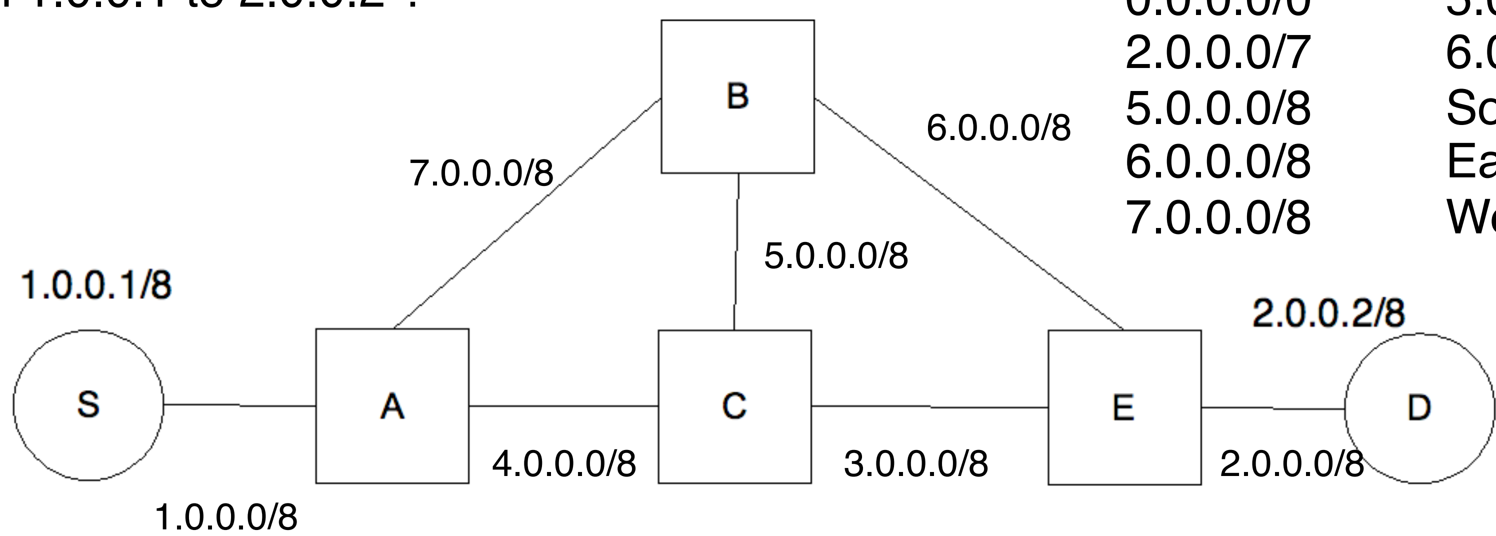
En pratique, le nexthop sera l'adresse IP d'un routeur, généralement directement joignable via la couche liaison de données, auquel le routeur local devra envoyer les paquets pour rejoindre un réseau distant.

Operation of an IP router (2)

- Operations performed for each packet
  1. Check whether the packet's destination address is one of the router's addresses
     - If yes, packet reached destination
  2. Query Forwarding Information Base that contains
     - list of directly connected networks with masks
     - list of reachable networks and intermediate router
  3. Lookup the most specific route in FIB
     - For each route A.B.C.D/M via Rx
     - compare M higher order bits of destination address with M higher order bits of routes to find longest match
     - forward packet along this route

IP Router : example

Path from 1.0.0.1 to 2.0.0.2 ?
and back



B's routing table

destination	interface/NH
0.0.0.0/0	5.0.0.C
2.0.0.0/7	6.0.0.E
5.0.0.0/8	South
6.0.0.0/8	East
7.0.0.0/8	West

A's routing table

destination	interface/NH
0.0.0.0/0	7.0.0.B
2.0.0.0/7	4.0.0.C
1.0.0.0/8	West
4.0.0.0/8	East
7.0.0.0/8	North

C's routing table

destination	interface/NH
1.0.0.0/8	5.0.0.B
2.0.0.0/7	3.0.0.E
2.0.0.0/8	5.0.0.B
3.0.0.0/8	East
4.0.0.0/8	West
5.0.0.0/8	North
6.0.0.0/7	5.0.0.B

E's routing table

destination	interface/NH
0.0.0.0/0	6.0.0.B
2.0.0.0/8	East
3.0.0.0/8	West
6.0.0.0/8	North

Handling IP packets in error

- Problem
  - What should a router/host do when it receives an errored packet
    - Example
      - Packet whose destination is not the current endhost
      - Packet containing a header with invalid syntax
      - Packet received with TTL=1
      - Packet destined to protocol not supported by host
- Solutions
  - Ignore and discard the errored packet
  - Send a message to the packet's source to warn it about the problem
    - ICMP : Internet Control Message Protocol
    - ICMP messages are sent inside IP packets by routers (mainly) and hosts
      - To avoid performance problems, most hosts/routers limit the amount of ICMP messages that they send

ICMP is defined in RFC792

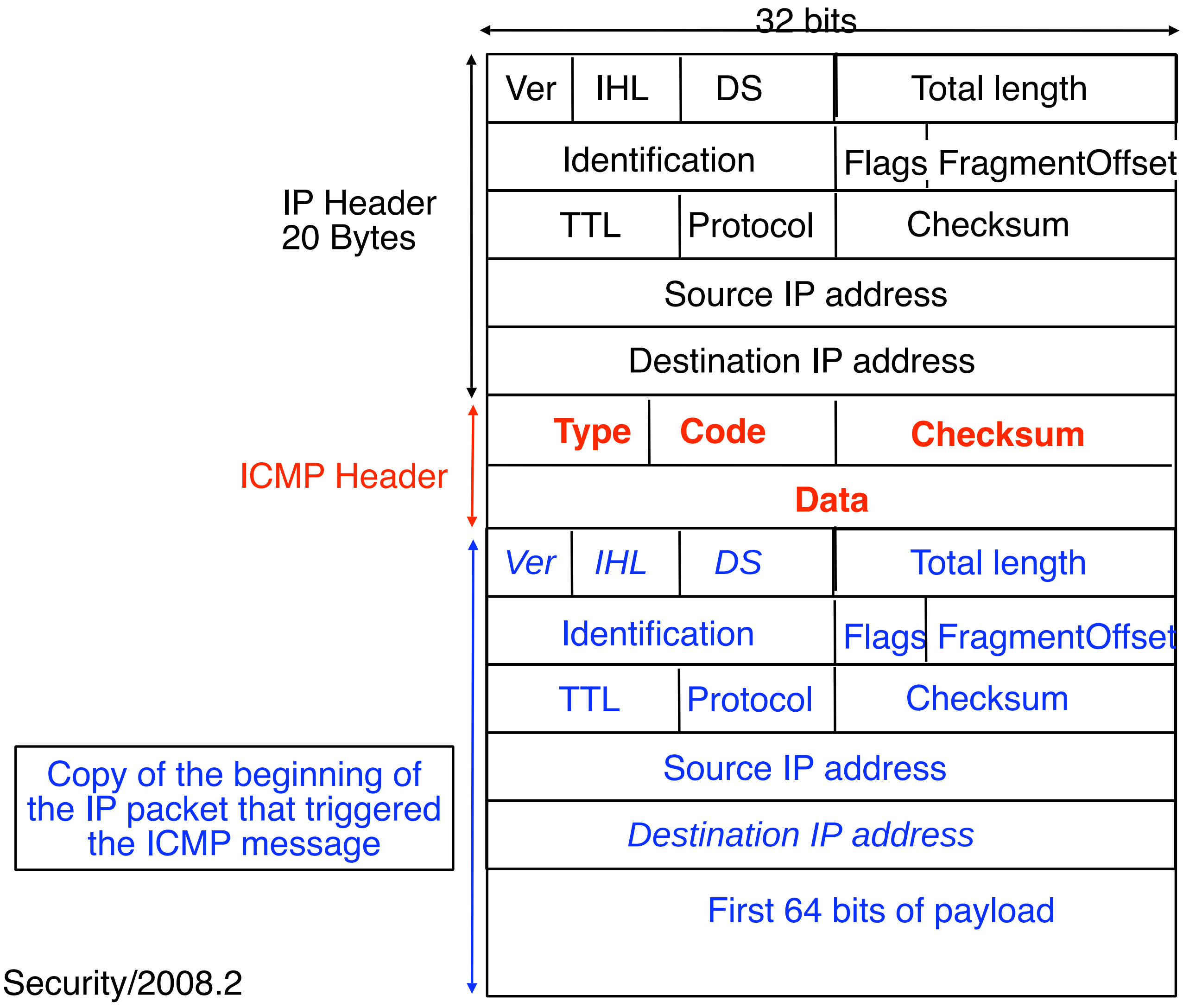
RFC792 Internet Control Message Protocol. J. Postel. Sep-01-1981.

Sample ICMP messages

- Routing error
  - Destination unreachable
    - Final destination of packet cannot be reached
      - Network unreachable for entire subnet
      - Host unreachable for an individual host
      - Protocol/Port unreachable for protocol/port on a reachable host
    - Redirect
      - The packet was sent to an incorrect first-hop router and should have been instead sent to another first-hop router
- Error in the IP header
  - Parameter Problem
    - Incorrect format of IP packet
  - TTL Exceeded
    - Router received packet with TTL=1
  - Fragmentation
    - the packet should have been fragmented, but its DF flag was true

ICMP

- Control message produced by a router or endsystem when a problem is detected



ICMP is defined in RFC792

A discussion of security attacks using ICMP can be found in
M. Baltatu, A. Lioy, F. Maino, D. Mazzocchi, Security issues in control, management and routing protocols, Computer Networks 34 (2000), 881-894

Usage of ICMP messages

- Examples
  - destination unreachable
    - the router sending this message did not have a route to reach the destination
  - time exceeded
    - the router sending the message received an IP packet with TTL=0
    - used by `traceroute`
  - redirect
    - to reach destination, another router must be used and ICMP message provides address of this router
  - echo request / echo reply
    - used by `ping`
  - fragmentation impossible
    - the packet should have been fragmented by the router sending the ICMP message by this packet had "Don't Fragment" set to true


```
 ping astrolabe
PING astrolabe (130.104.229.109) 56(84) bytes of data.
64 bytes from astrolabe (130.104.229.109): icmp_seq=1 ttl=245 time=20.7 ms
64 bytes from astrolabe (130.104.229.109): icmp_seq=2 ttl=245 time=20.2 ms
64 bytes from astrolabe (130.104.229.109): icmp_seq=3 ttl=245 time=20.1 ms

--- astrolabe ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2016ms
rtt min/avg/max/mdev = 20.156/20.383/20.722/0.244 ms
```

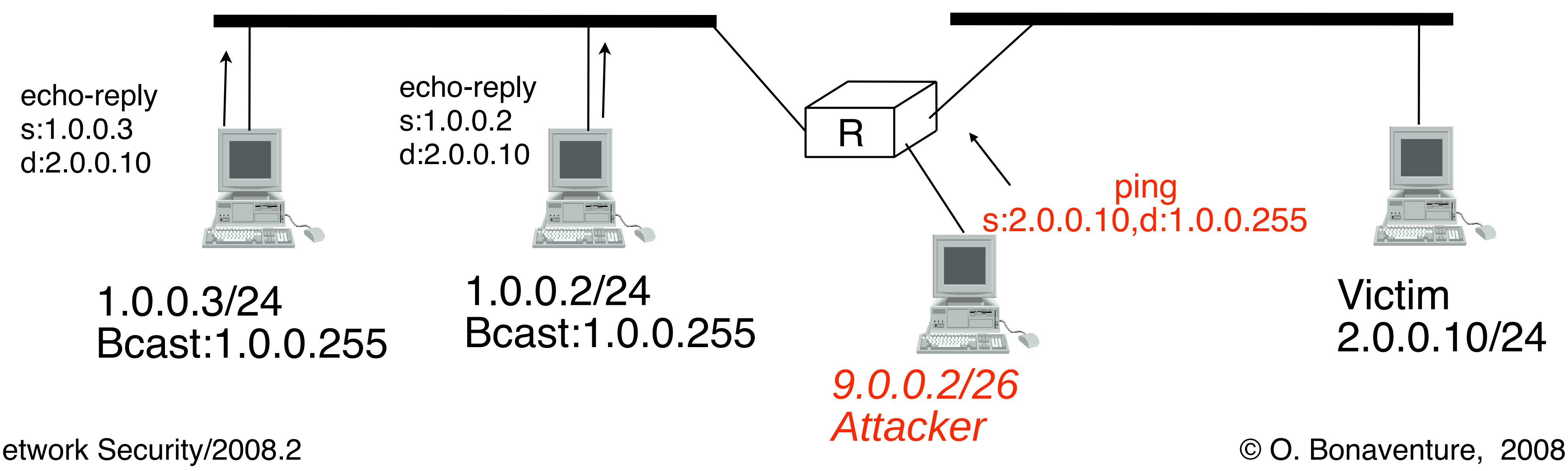
Exemple de traceroute

```
] traceroute www.geant.net
traceroute: Warning: ckecksums disabled
traceroute to newweb.dante.org.uk (62.40.101.34), 30 hops max, 40 byte packets
 1  accelar-1 (130.104.229.126)  1.890 ms  1.752 ms  1.723 ms
 2  XVLX-CR.fsa.ucl.ac.be (130.104.233.233)  1.620 ms  1.620 ms  1.603 ms
 3  CsPythagore.sri.ucl.ac.be (130.104.254.221)  1.317 ms  1.305 ms  1.302 ms
 4  CsHalles.sri.ucl.ac.be (130.104.254.201)  1.512 ms  1.425 ms  1.415 ms
 5  193.191.11.9 (193.191.11.9)  0.891 ms  0.780 ms  0.780 ms
 6  193.191.1.197 (193.191.1.197)  1.166 ms  1.263 ms  1.079 ms
 7  193.191.1.2 (193.191.1.2)  1.329 ms  1.107 ms  1.100 ms
 8  belnet.be1.be.geant.net (62.40.103.13)  1.341 ms  1.490 ms  1.323 ms
 9  be.nl1.nl.geant.net (62.40.96.22)  4.779 ms  4.586 ms  4.515 ms
10  nl.uk1.uk.geant.net (62.40.96.182)  12.259 ms  12.051 ms  12.029 ms
11  62.40.101.34 (62.40.101.34)  12.811 ms  12.310 ms  12.645 ms
```

Security risks with ICMP `echo request`

- Echo request
  - ICMP message type 1
  - A host receiving this message should reply by sending ICMP message with type 8 (echo reply)

- Smurf attack
  - Send spoofed ICMP echo reply to broadcast addr



The smurf attack was popular a few years ago. On many networks, the broadcast address is either the address “.0” or “.255”.

To limit the security risks with ICM echo requests messages, many enterprise networks and ISPs have implemented filters to limit the amount of ICMP echo request messages that enter their network. Some hosts are also configured by default to avoid replying to ICMP echo requests sent to the broadcast address and also limit the rate of accepted and generated ICMP echo messages.

The echo request and echo reply ICMP messages are used by ping.
RFC792 also defined two other ICMP messages to obtain information about a remote host :
- timestamp and timestamp reply
- information request and information reply

Those two types of messages can reveal information about the endsystem to a distant attacker. Security guidelines usually recommend to disable such ICMP messages.

Security risks with ICMP `destination unreachable`

- Utilisation
  - Sent by a router to indicate
    - IP address is (temporarily ?) unreachable from router
    - Packet with DF bit set should have been fragmented
      - Data contains MTU to be used
  - Sent by endsystem to indicate
    - UDP/TCP port is (temporarily ?) unreachable

- Upon reception
  - ICMP message passed to transport layer
    - should check IP header and transport header of ICMP message

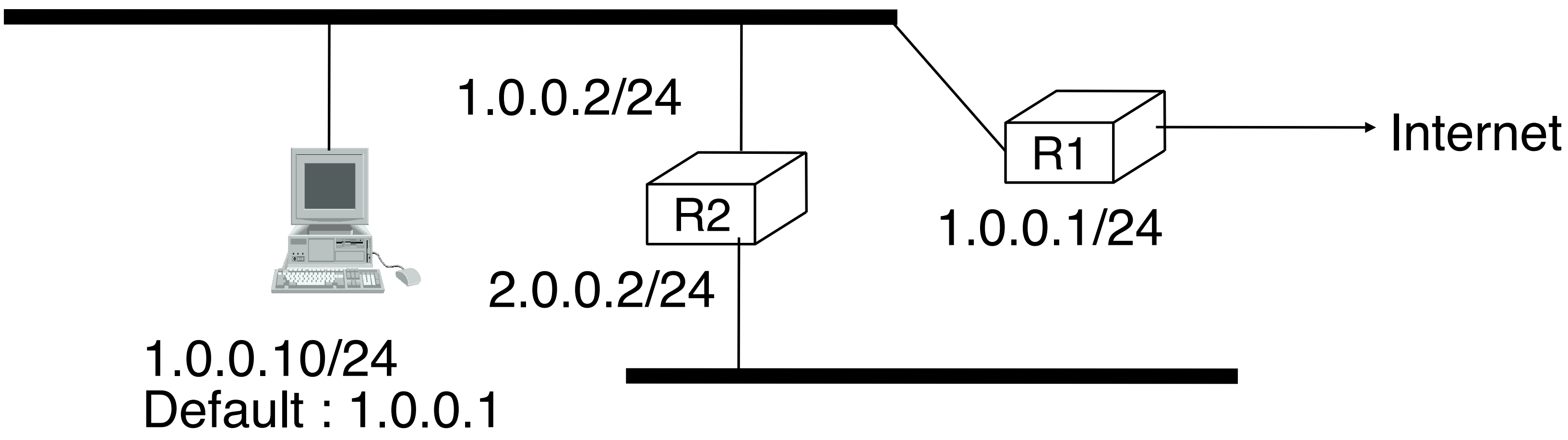
- Security risks
  - Transport layer or application could stop upon reception of such a message
  - Could be used to force sender to use small MTU

To be successful with such an attack, the attack needs to guess the source, destination IP addresses and the source and destination port numbers. As the ICMP message contains the first 64 bits of the segment contained in the IP packet that caused the error, it would be possible for a TCP implementation to check that the last 32 bits of the ICMP message correspond to a valid sequence number.

Note that blocking ICMP messages on a firewall is a bad solution if the TCP implementation always sends packets with the DF flag set. Without the ICMP messages, it might be impossible to exchange packets over a TCP connection if there are paths with a lower MTU between the sender and the destination.
A similar risk of reduction in transmission rate occurs with the ICM source quench message. This message could be sent by a router to indicate that its buffers were full. Most routers do not use this message any more and it is deprecated, but TCP implementations usually respond to such messages by halving their congestion window.
The time exceeded message is less problematic. It is sent only when a packet was received with TTL=0 or when a packet could not be reassembled by the destination host. TCP implementations usually do not react to such messages.

Security risks with ICMP `route redirect`

- Utilisation
  - Used by routers to inform hosts that they should use another router to reach a destination



- Security risks
  - Attacker could force victim to use him as the router to reach important destinations -> MITM
  - Attacker could force victim to use non-existing router to reach important prefixes -> DoS

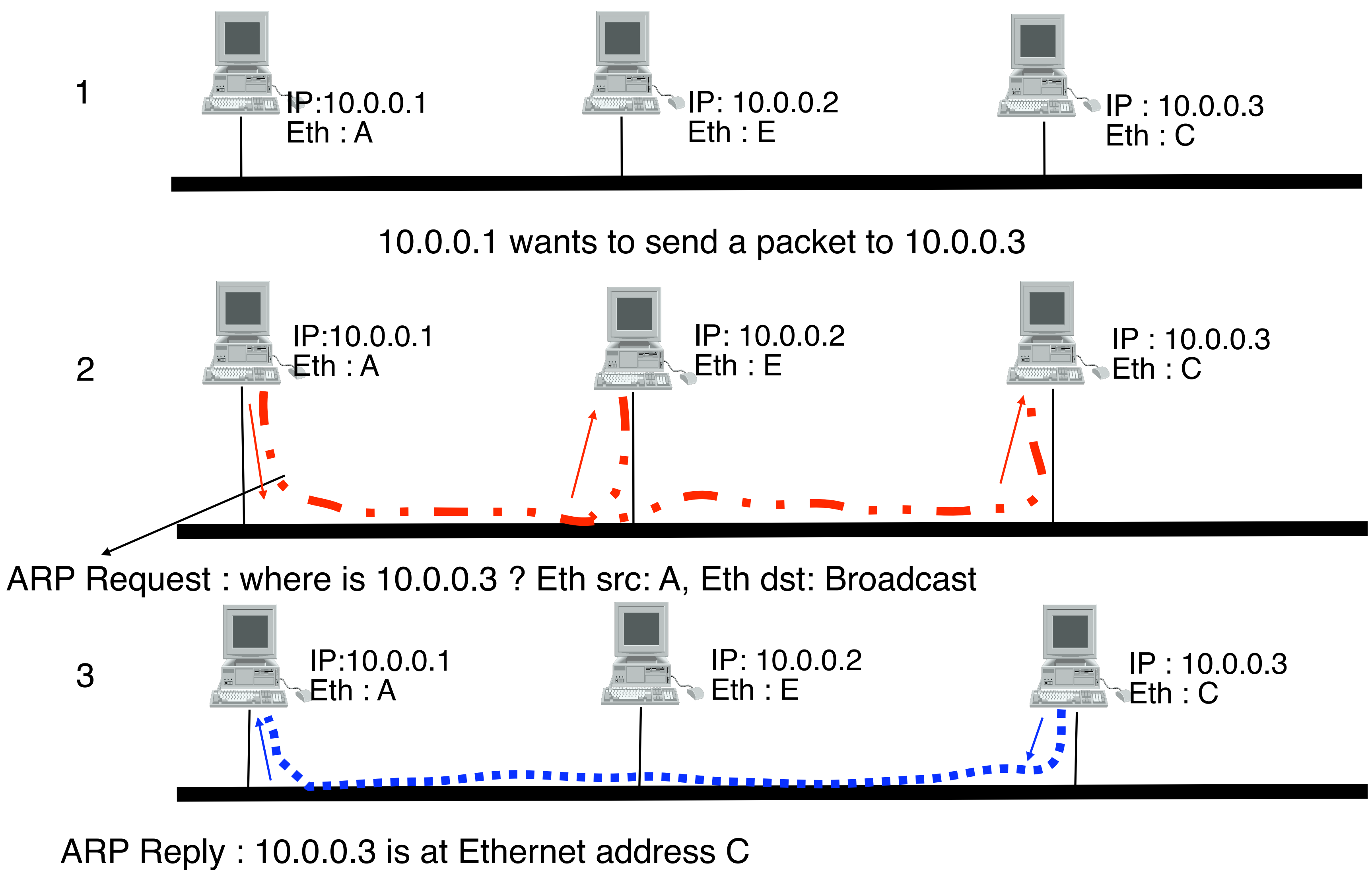
For the MITM attack, the attacker must be present on the same LAN (or VLAN) as the victim, but for the DoS attack it only needs to send a spoofed packet to the victim to force him to install an invalid route inside its routing table.

For those reasons, ICMP route redirect messages should be considered with great care. In practice, it would be better to avoid them as there are few LANs containing both endsystems and routers.

If a LAN must contain both endsystems and routers, then from a security viewpoint, a better solution is to utilize non-optimal routing, i.e. configure the routers to never generate ICMP route redirect messages

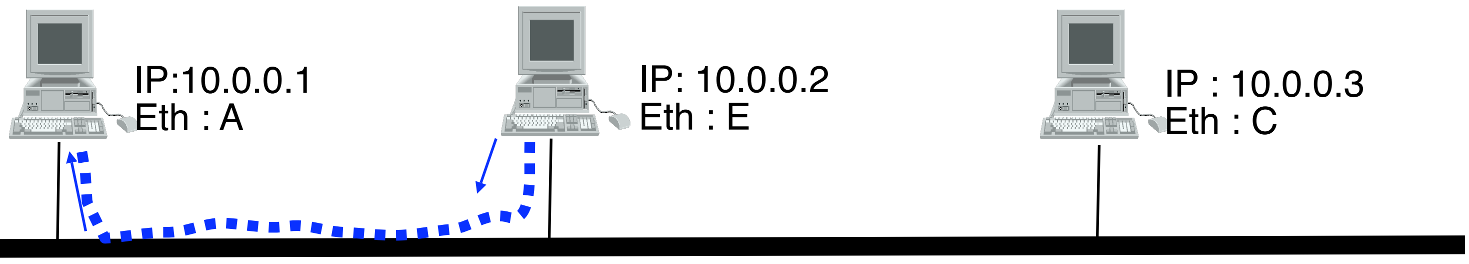
Address Resolution Protocol

- ARP

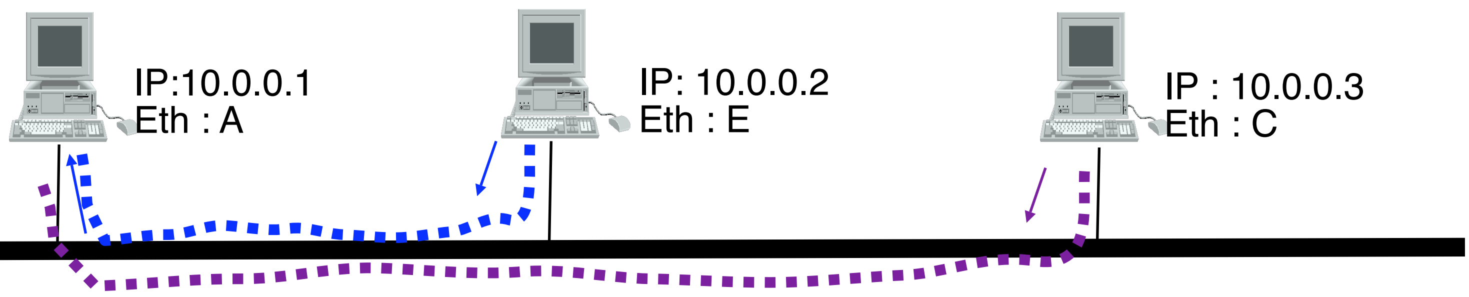


Security issues with ARP

- What happens if ?



ARP Reply : 10.0.0.2 is at Ethernet address E !



ARP Reply : 10.0.0.2 is at Ethernet address E !

ARP Reply : 10.0.0.2 is at Ethernet address C !

- Some TCP/IP implementations perform a ARP request for their own IP address when booting to detect misconfigurations

Internet and Network security

- Crypto building blocks
- Application-layer security
- Transport-layer security
- Network-layer security
  - IPv4
  - IPv6
  - IPSec
  - Routing security

IP version 6

- Outline
  - Motivations for IP version 6
  - IPv6 addressing architecture
  - IPv6 packets
  - ICMP v6

There are many books and information about IPv6

An interesting book, but written in French, is G. Cizault, IPv6 Théorie et Pratique, O Reilly
The new versions of this book are available online : http://livre.point6.net/index.php/Accueil

A more practically oriented book is
I. van Beijnum, Running IPv6, APress, 2006

IPv6 standardisation is carried out within the IETF, http://www.ietf.org

Other resources include

P. Smith, Introduction to IPv6, NANOG 42, ftp://ftp-eng.cisco.com/pfs/seminars/NANOG42-IPv6-Introduction.pdf

http://www.6journal.org/

http://www.ist-ipv6.org/

Information about IPv6 aware software and hardware is available from

http://www.ipv6-to-standard.org/

Issues with IPv4

- Late 1980s
  - Exponential growth of Internet
- 1990
  - Other network protocols exist
  - Governments push for CLNP
- 1992
  - Most class B networks have been assigned
  - Class based routing failure
  - Networking experts warn that IPv4 address space could become exhausted

For more information about the exhaustion of IPv4 addresses, see
http://www.potaroo.net/tools/ipv4/index.html

Issues with IPv4 (2)

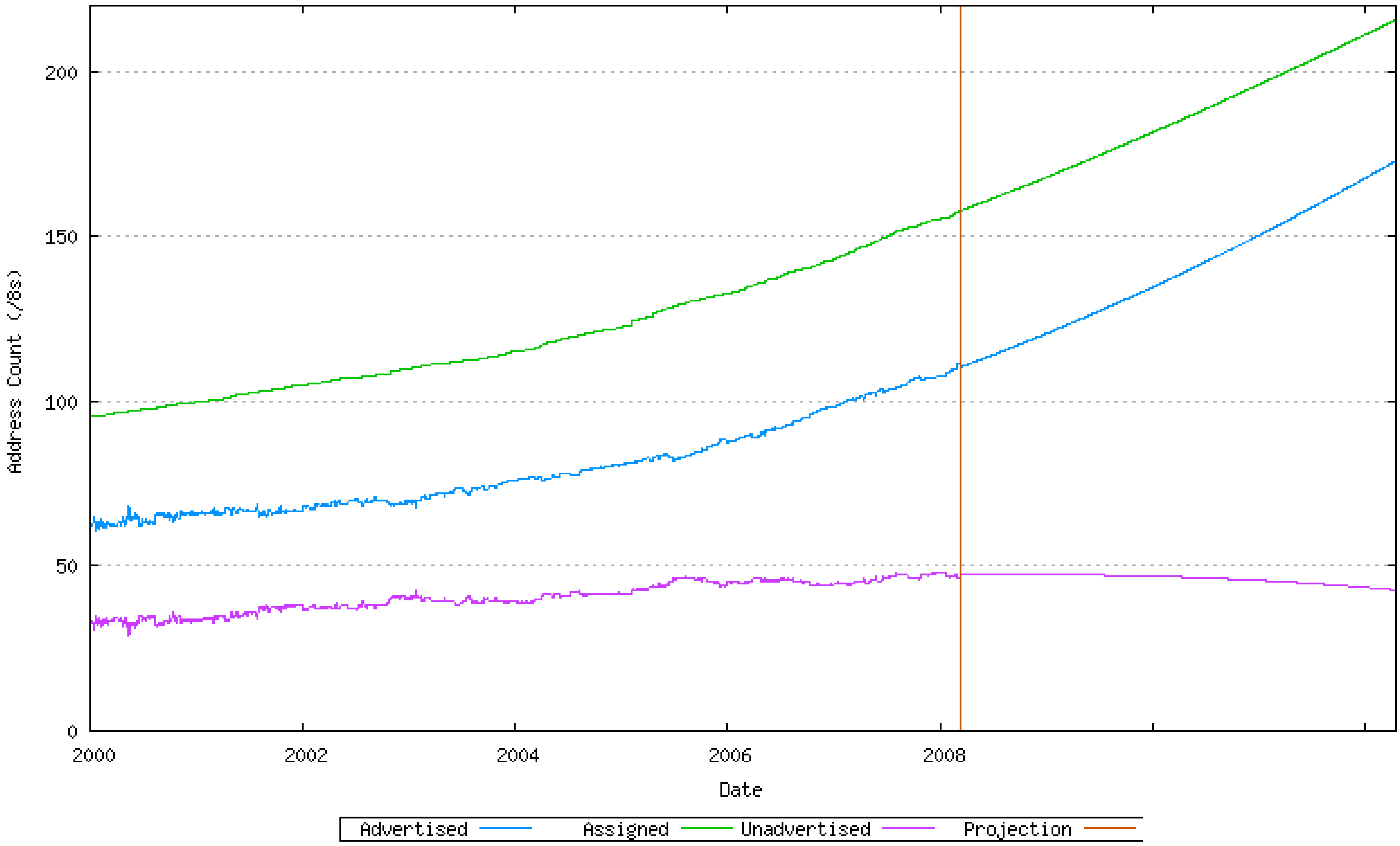
- How to solve the exhaustion of class B addresses ?
- Short term solution
  - Define Classless Interdomain Routing (CIDR) and introduce the necessary changes in routers
  - Deployment started in 1994
- Long term solution
  - Develop Internet Protocol - next generation (IPng)
    - call for proposals RFC1550, Dec 1993
    - Criteria for choix, RFC1719 and RFC1726, Dec. 1994
    - Proposed solutions
      - TUBA - RFC1347, June 1992
      - PIP – RFC1621, RFC1622, May 1994
      - CATNIP – RFC1707, October 1994
      - SIP – RFC1710, October 1994
      - NIMROD – RFC1753, December 1994
      - ENCAPS – RFC1955, June 1996

Issues with IPv4 (3)

- Implementation issues - 1990s
  - IPv4 packet format is complex
  - IP forwarding is difficult in hardware

- Missing functions - 1990s
  - IPv4 requires lots of manual configuration
    - Competing protocols (CLNP, Appletalk, IPX, ...) already supported autoconfiguration in 1990s
  - How to support Quality of Service in IP ?
    - Integrated services and Differentiated services did not exist then
  - How to better support security in IP ?
    - Security problems started to appear but were less important than today
  - How to better support mobility in IP ?
    - GSM started to appear and some were dreaming of mobile devices attached to the Internet

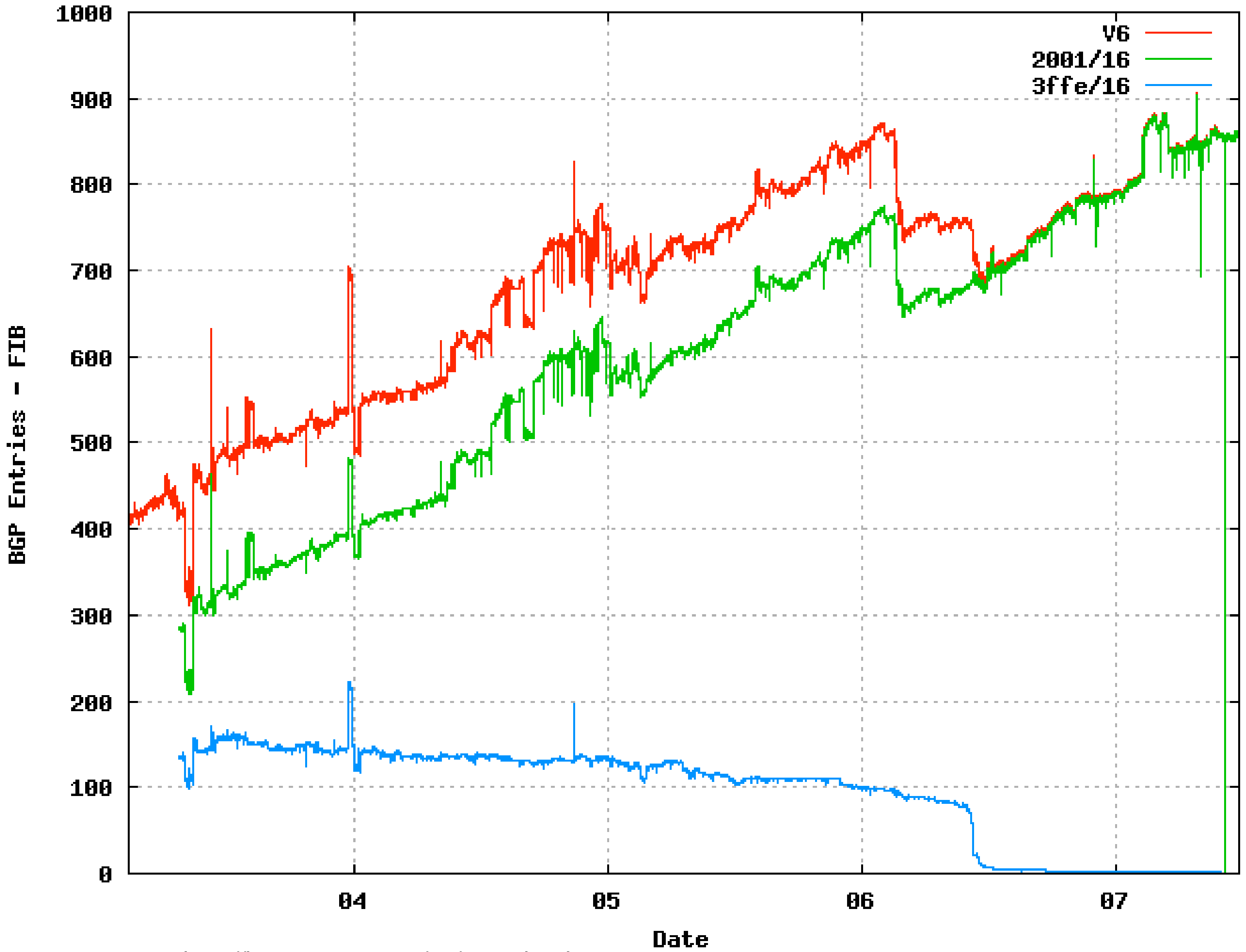
Main motivation today
IPv4 address exhaustion



Source http://www.potaroo.net/tools/ipv4/index.html

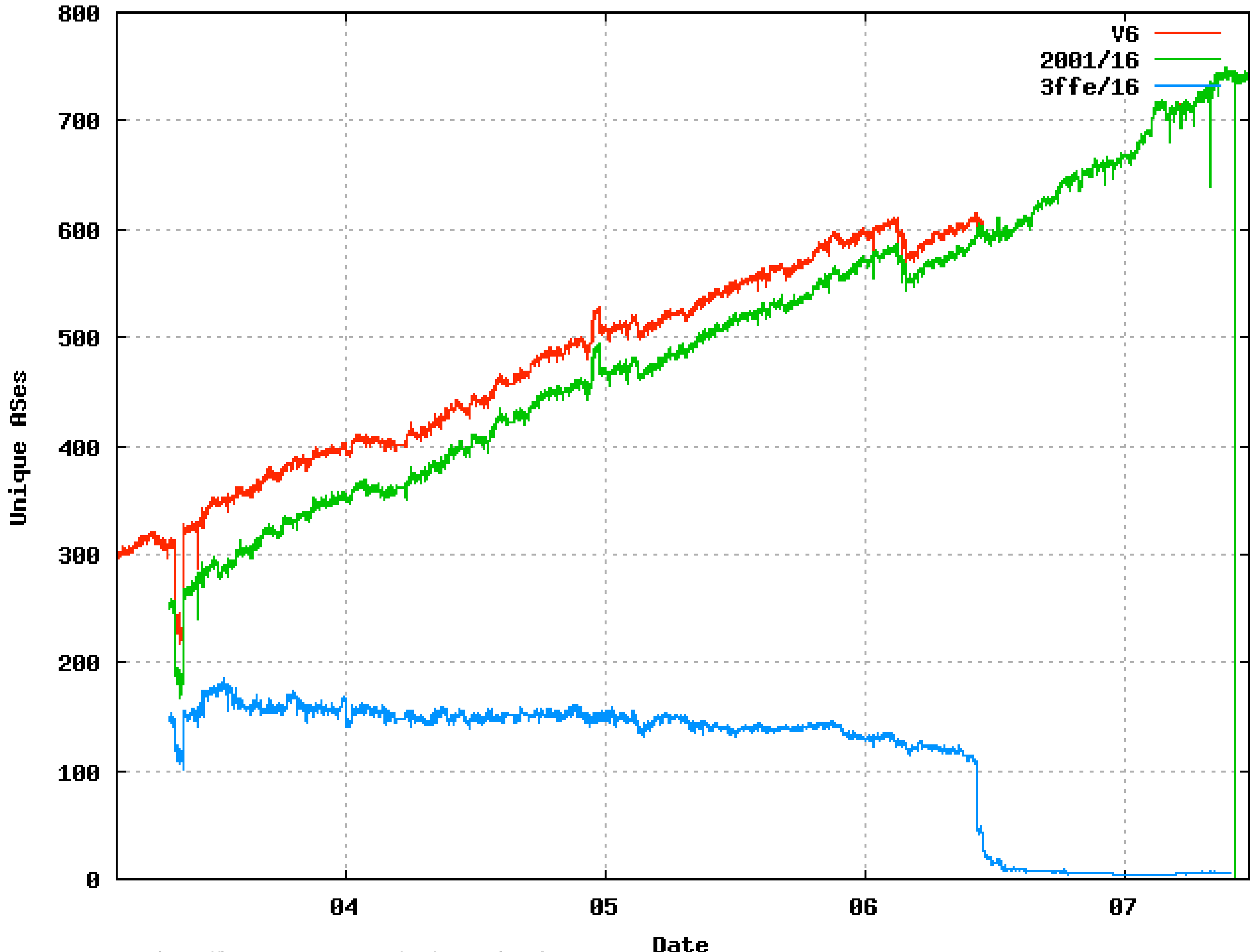
This figure shows the number of IPv4 prefixes used on the global Internet. In addition, some networks, e.g. large cable networks, have had difficulties in using IPv4 due to the limited number of available addresses. For example, comcast is planning to use IPv6 to manage its cable modems mainly because IPv4 does not allow them to have enough addresses to identify all their potential cable modems in a scalable manner, see http://www.nanog.org/mtg-0606/durand.html

IPv6 usage
advertised prefixes



source http://bgp.potaroo.net/v6/v6rpt.html

Current IPv6 usage ASes using IPv6



source http://bgp.potaroo.net/v6/v6rpt.html

In contrast, the number of ASes using IPv4 is much larger. In March 2008, more than 27000 ASes were advertising IPv4 addresses, see http://bgp.potaroo.net/bgprpts/rva-index.html

Can we avoid deploying IPv6 by using NAT ?

- Network address translation
- Benefits
  - Reduces consumption of public IPv4 addresses
  - “Hides” internal IPv4 addresses inside homes and corporate networks
- Drawbacks
  - Breaks the end-to-end principle
  - Intermediate nodes may modify packet content
    - IP addresses
    - TCP/UDP port information
    - Some protocols encode IP addresses inside payload
      - ftp
      - ...

For a detailed discussion of NAT and its implications, see :

[RFC2993] Hain, T., "Architectural Implications of NAT", RFC 2993, November 2000.

[RFC3027] Holdrege, M. and P. Srisuresh, "Protocol Complications with the IP Network Address Translator (NAT)", RFC 3027, January 2001.

[RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

IP version 6

- Outline
  - Motivations for IP version 6
  - IPv6 addressing architecture
  - IPv6 packets
  - ICMP v6

IPv6 addresses

IPv4

IP version 6

- Each IPv6 address is encoded in 128 bits
  - 3.4 x 10^38 possible addressable devices
    - 340,282,366,920,938,463,463,374,607,431,768,211,456
  - ~ 5 x 10^28 addresses per person on the earth
  - 6.65 x 10^23 addresses per square meter
  - Looks unlimited.... today
- Why 128 bits ?
  - Some wanted variable size addresses
    - to support IPv4 and 160 bits OSI NSAP
  - Some wanted 64 bits
    - Efficient for software, large enough for most needs
  - Hardware implementers preferred fixed size

IP version 4 supports 4,294,967,296 distinct addresses, but some are reserved for :
private addresses (RFC1918)
loopback (127.0.0.1)
multicast
...

The IPv6 addressing architecture

- Three types of IPv6 addresses
  - Unicast addresses
    - An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address
  - Anycast addresses
    - An identifier for a set of interfaces. A packet sent to an
    - anycast address is delivered to the “nearest” one of the interfaces identified by that address
  - Multicast addresses
    - An identifier for a set of interfaces. A packet sent to a multicast address is delivered to all interfaces identified by that address.

The IPv6 addressing architecture is defined in :

R. Hinden, S. Deering, IP Version 6 Addressing Architecture, RFC4291, February 2006

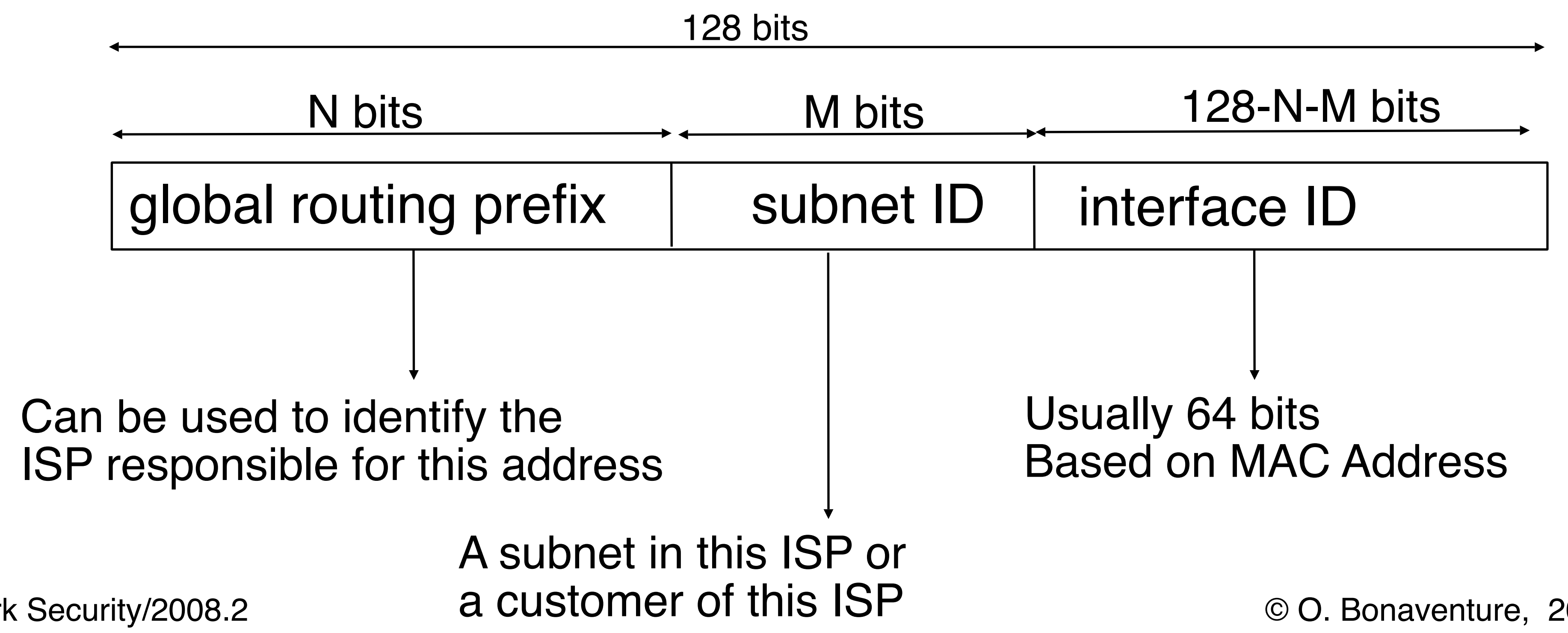
Representation of IPv6 addresses

- How can we write a 128 bits IPv6 address ?
  - Hexadecimal format
    - FEDC:BA98:7654:3210:FEDC:BA98:7654:3210
    - 1080:0:0:0:8:800:200C:417A
  - Compact hexadecimal format
    - Some IPv6 addresses contain lots of zero
      - utilize "::" to indicate one or more groups of 16 bits of zeros. The "::" can only appear once in an address
      - Examples
        - 1080:0:0:0:8:800:200C:417A = 1080::8:800:200C:417A
        - FF01:0:0:0:0:0:0:101 = FF01::101
        - 0:0:0:0:0:0:0:1 = ::1

The IPv6 unicast addresses

- Special addresses
  - Unspecified address : 0:0:0:0:0:0:0:0
  - Loopback address : 0:0:0:0:0:0:0:1

- Global unicast addresses
  - Addresses will be allocated hierarchically



Today, the default encoding for global unicast addresses is to use :

48 bits for the global routing prefix (first three bits are set to 001)

16 bits for the subnet ID

64 bits for the interface ID

Allocation of IPv6 addresses

- IANA controls all IP addresses and delegates assignments of blocks to Regional IP Address Registries (RIR)
  - RIPE, ARIN, APNIC, AFRINIC, ...
- An organisation can be allocated two different types of IPv6 addresses
  - Provider Independent (PI) addresses
    - Usually allocated to ISPs or very large enterprises directly by RIRs
    - Default size is /32
  - Provider Aggregatable (PA) addresses
    - Smaller prefixes, assigned by ISPs from their PI block
    - Size
      - /48 in the general case, except for very large subscribers
      - /64 when t one and only one subnet is needed by design
      - /128 when it is absolutely known that one and only one device is connecting.

See http://www.ripe.net/ripe/docs/ripe-388.html for the policy used by RIPE to allocate IP prefixes in Europe

The IPv6 link-local addresses

- Used by hosts and routers attached to the same LAN to exchange IPv6 packets when they don’t have/need globally routable addresses

128 bits

10 bits	54 bits	64 bits
FE80	0000000000.....00000000000	interface ID

- Each host must generate one link local address for each of its interfaces
  - Each IPv6 host will use several IPv6 addresses
- Each routers must generate one link local address for each of its interfaces

Site-local addresses were defined in the first IPv6 specifications, but they are now deprecated and should not be used.

Recently “private” addresses have been defined as Unique Local IPv6 Addresses as a way to allow entreprise to obtain IPv6 addresses without being forced to request them from providers or RIRs.

The way to choose such a ULA prefix is defined in :

R. Hinden, B. Haberman, Unique Local IPv6 Unicast Addresses, RFC4193, October 2005

Recently, the case for a registration of such addresses has been proposed, see :
R. Hinden, G. Huston, T. Narten, Centrally Assigned Unique Local IPv6 Unicast Addresses, internet draft, <draft-ietf-ipv6-ula-central-02.txt>, work in progress, June 2007

See also

http://www.ripe.net/ripe/policies/proposals/2007-05.html -

The IPv6 anycast addresses

- Definition
  - An IPv6 anycast address is an address that is assigned to more than one interface (typically belonging to different nodes), with the property that a packet sent to an anycast address is routed to the "nearest" interface having that address, according to the routing protocols' measure of distance.
- Usage
  - Multiple redundant servers using same address
  - Example DNS resolvers and DNS servers
- Representation
  - IPv6 anycast addresses are unicast addresses
  - Required subnet anycast address

n bits	128-n bits
IPv6 subnet prefix	00000000000000000000000

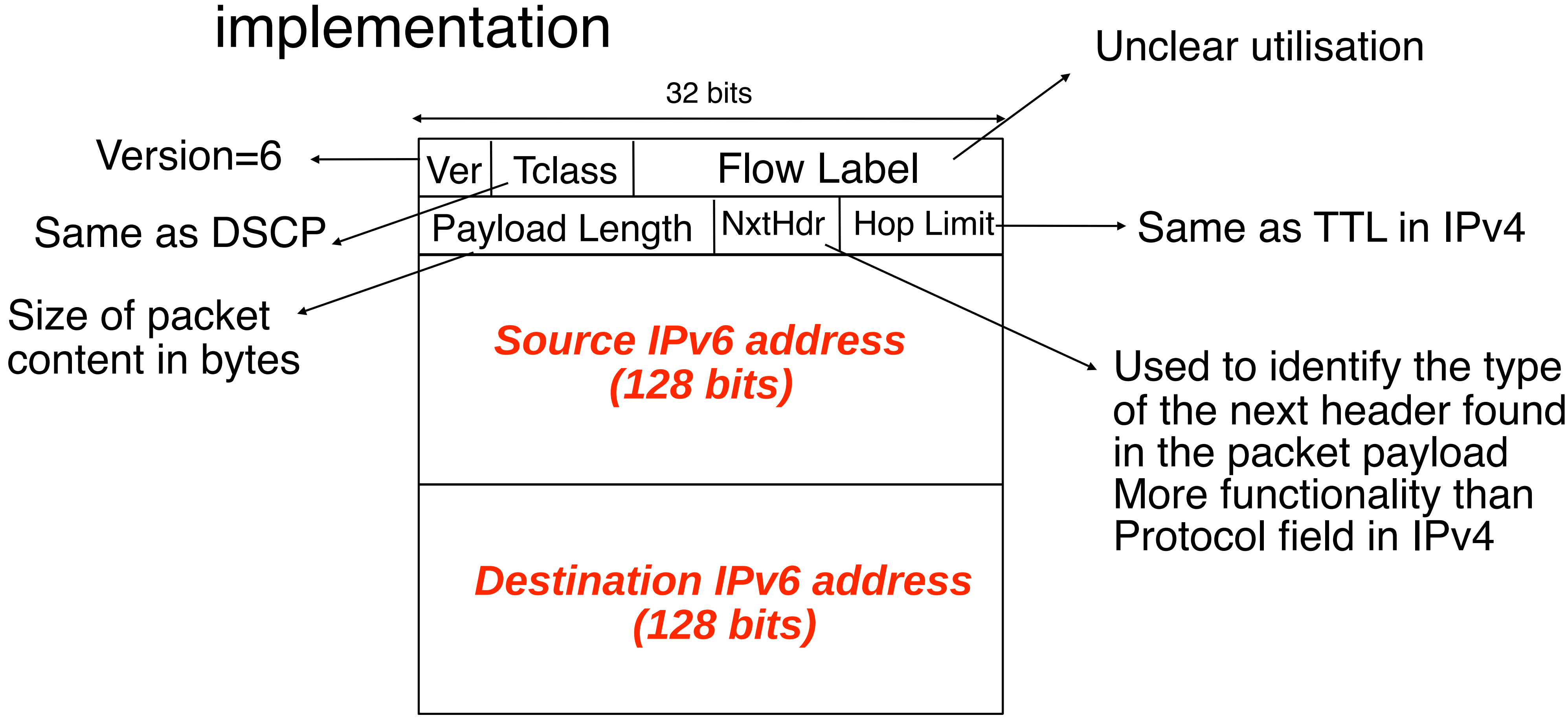
The allocated anycast addresses are references in http://www.iana.org/assignments/ipv6-anycast-addresses

IP version 6

- Outline
  - Motivations for IP version 6
  - IPv6 addressing architecture
  - IPv6 packets
  - ICMP v6

The IPv6 packet format

- Simplified packet format
  - Fields aligned on 32 bits boundaries to ease implementation



- No checksum in IPv6 header
  - rely on datalink and transport checksums

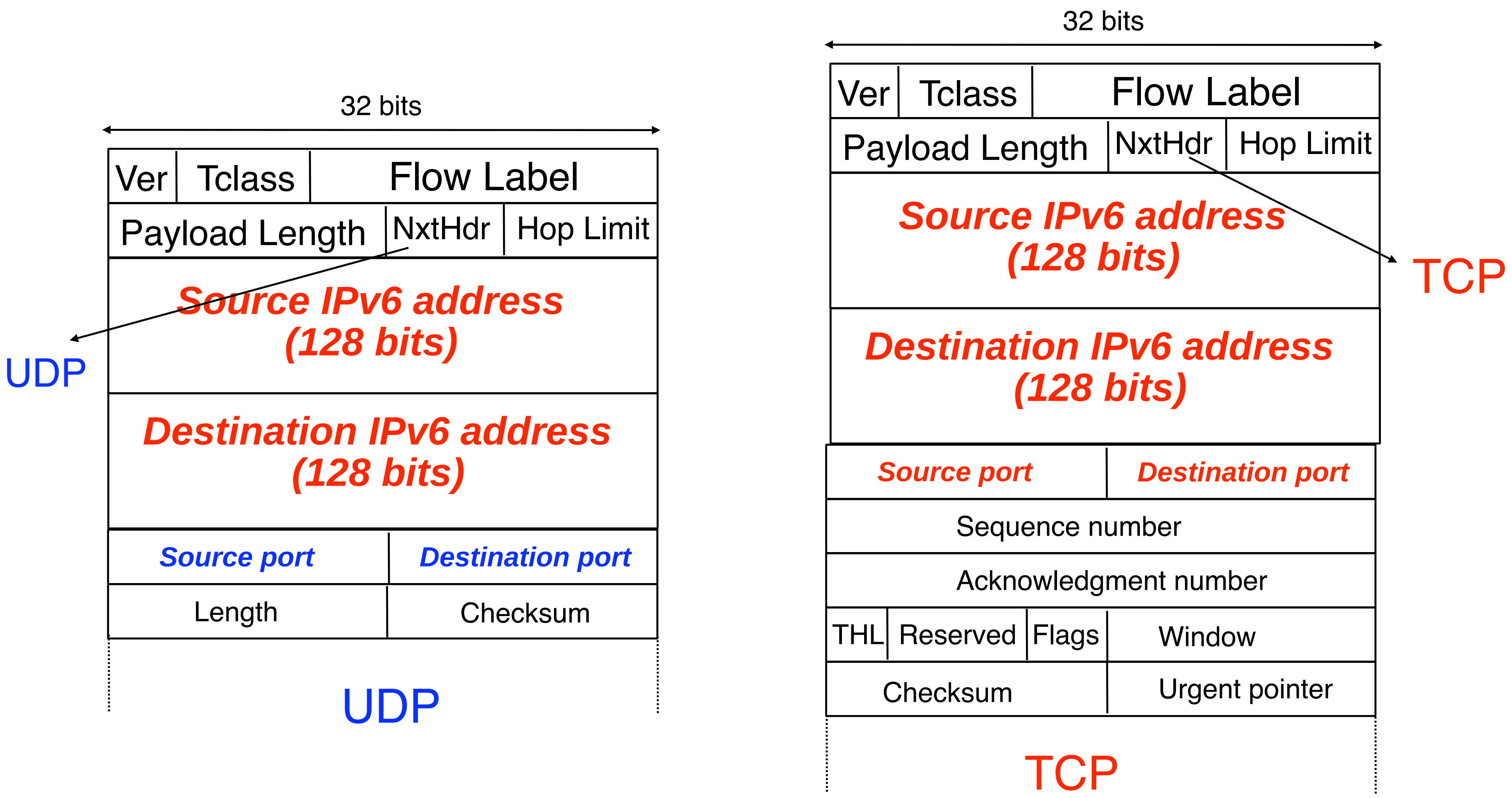
The IPv6 packet format is described in
S. Deering, B. Hinden, Internet Protocol, Version 6 (IPv6) Specification , RFC2460, Dec 1998

Several documents have been written about the usage of the Flow label. The last one is

J. Rajahalme, A. Conta, B. Carpenter, S. Deering, IPv6 Flow Label Specification, RFC3697, 2004

However, this proposal is far from being widely used and deployed.

Sample IPv6 packets



- Identification of a TCP connection
  - IPv6 source, IPv6 destination, Source and Destination ports

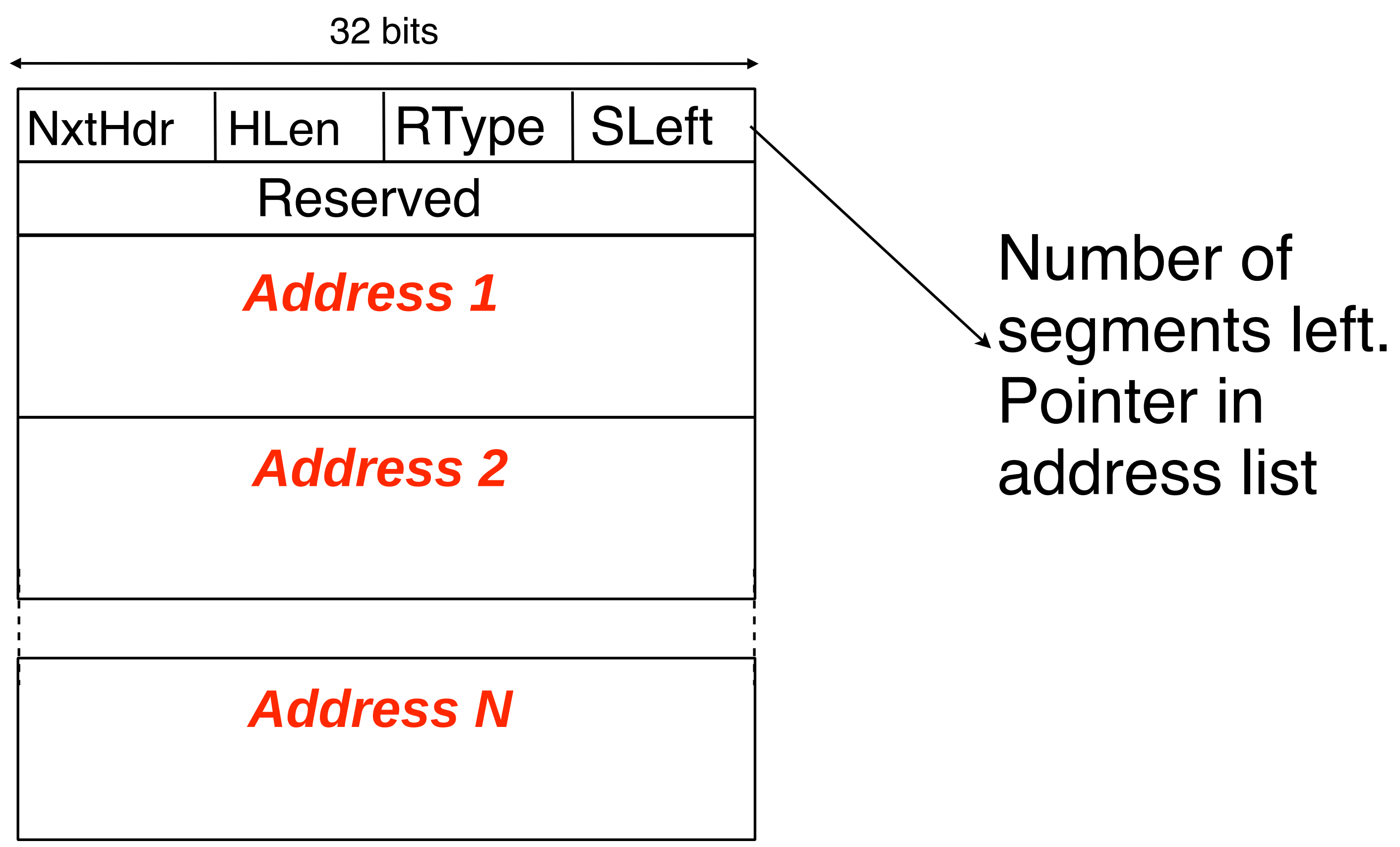
IPv6 does not require changes to TCP and UDP for IPv4. The only modification is the computation of the checksum field of the UDP and TCP headers since this checksum is computed by concerning a pseudo header that contains the source and destination IP addresses.

The IPv6 extension headers

- Several types of extension headers
  - Hop-by-Hop Options
    - contains information to be processed by each hop
  - Routing (Type 0 and Type 2)
    - contains information affecting intermediate routers
  - Fragment
    - used for fragmentation and reassembly
  - Destination Options
    - contains options that are relevant for destination
  - Authentication
    - for IPSec
  - Encapsulating Security Payload
    - for IPSec
- Each header must be encoded as n*64 bits

An example hop-by-hop option is the router alert option defined in
A. Jackson, C. Partridge, IPv6 Router Alert Option RFC2711, 1999

Type 0 Routing header

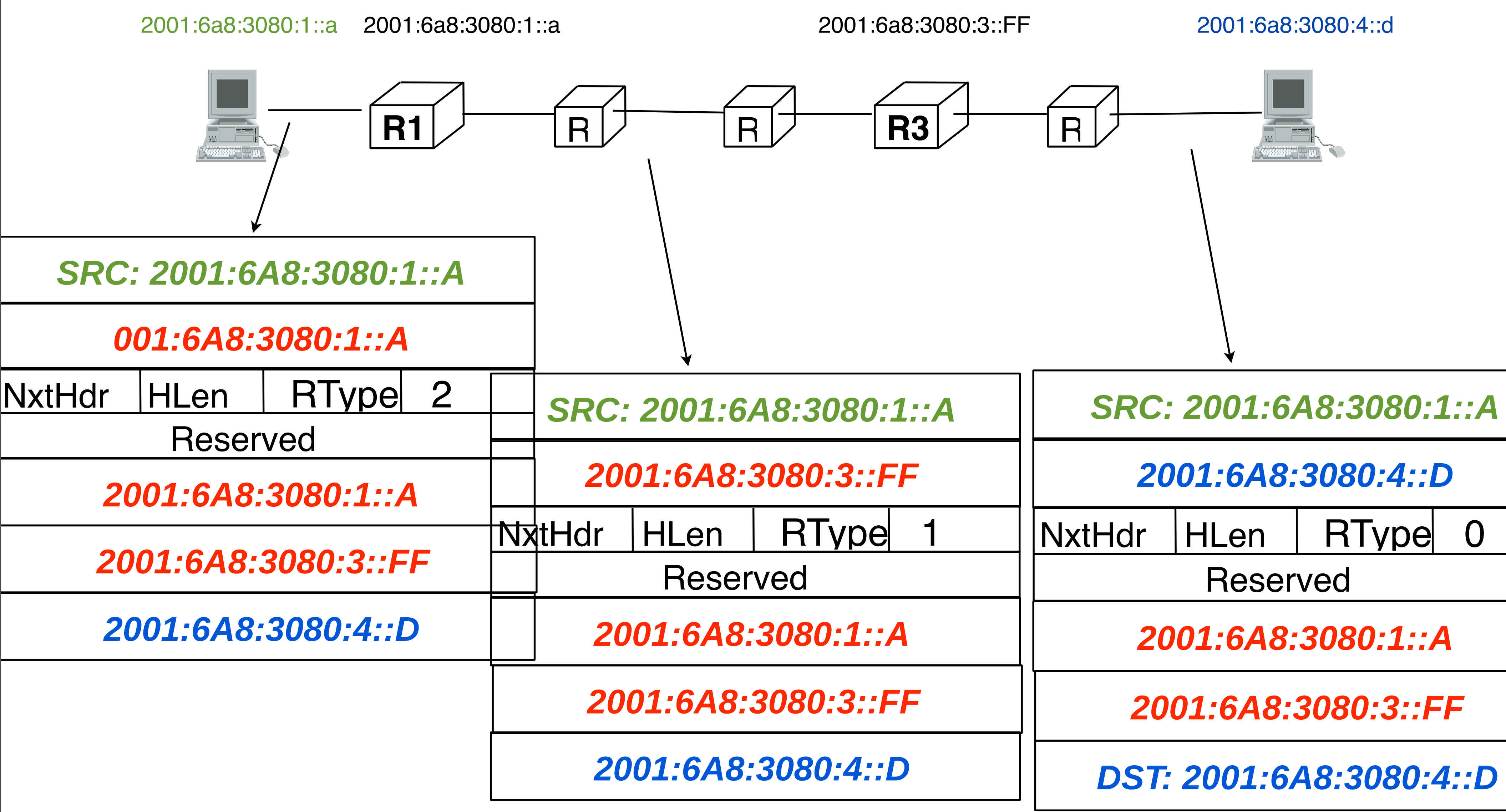


- Defined as "*a mean for a source to list one or more intermediate nodes to be "visited" on the way to a packet's destination*

The Type 0 Routing header is specified in RFC2460

Two other types of routing headers have been defined. Type 1 is experimental and never used. Type 2 is specific for Mobile IPv6 that will be covered later.

Type 0 routing header example



Issues with Type 0 Routing header

- Type 0 RH is a generalisation of IPv4 source routing

- The IPv6 specification is unclear about the processing of Type 0 RH
  - Node = *a device that implements IPv6*
  - Router = *a node that forwards IPv6 packets not explicitly addressed to itself*
  - Host = *any node that is not a router*

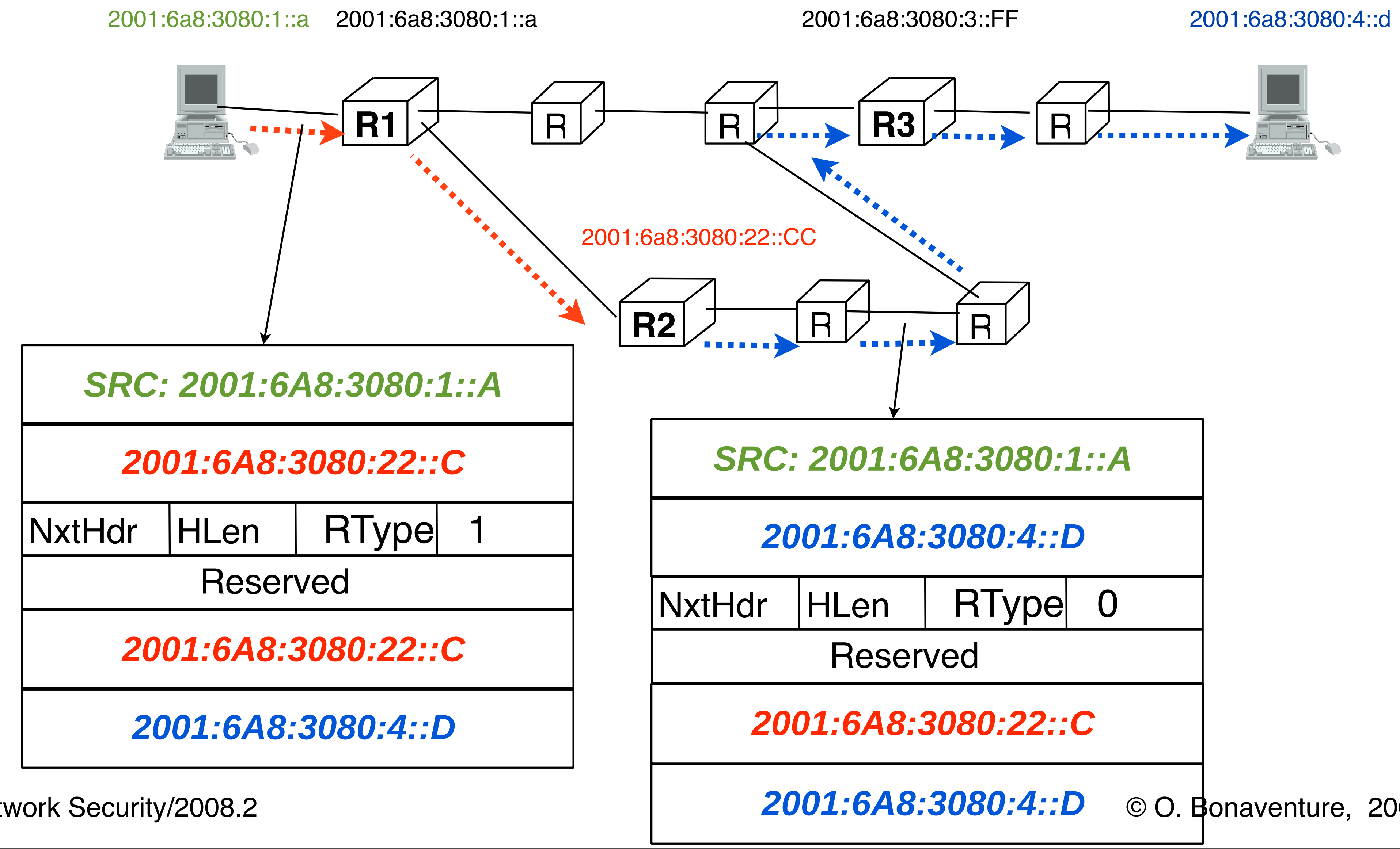
- How to process headers ?
  - *IPv6 nodes must accept and attempt to process extension headers in any order and occurring any number of times in the same packet, . . .*

The type 0 routing header was deprecated in
J. Abley, P. Savola, G. Neville-Neil, Deprecation of Type 0 Routing Headers in IPv6 RFC5095, Dec. 2007

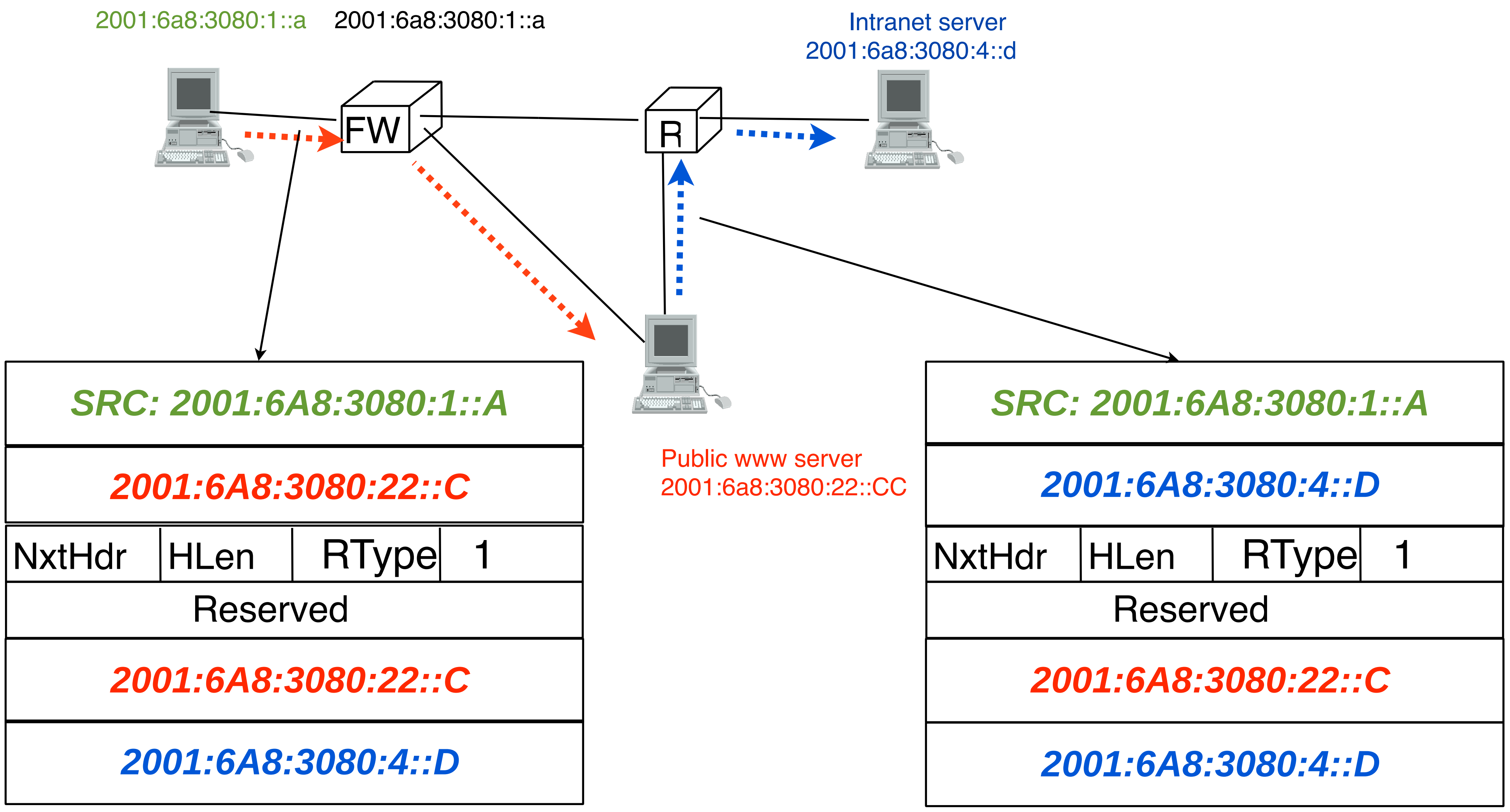
For more information about the security issues with this header, see
Biondi, P. and A. Ebalard, "IPv6 Routing Header Security", CanSecWest Security Conference 2007, April 2007. http://www.secdev.org/conf/IPv6_RH_security-csw07.pdf

Other usage of Type 0 RH

- Improved topology discovery with traceroute

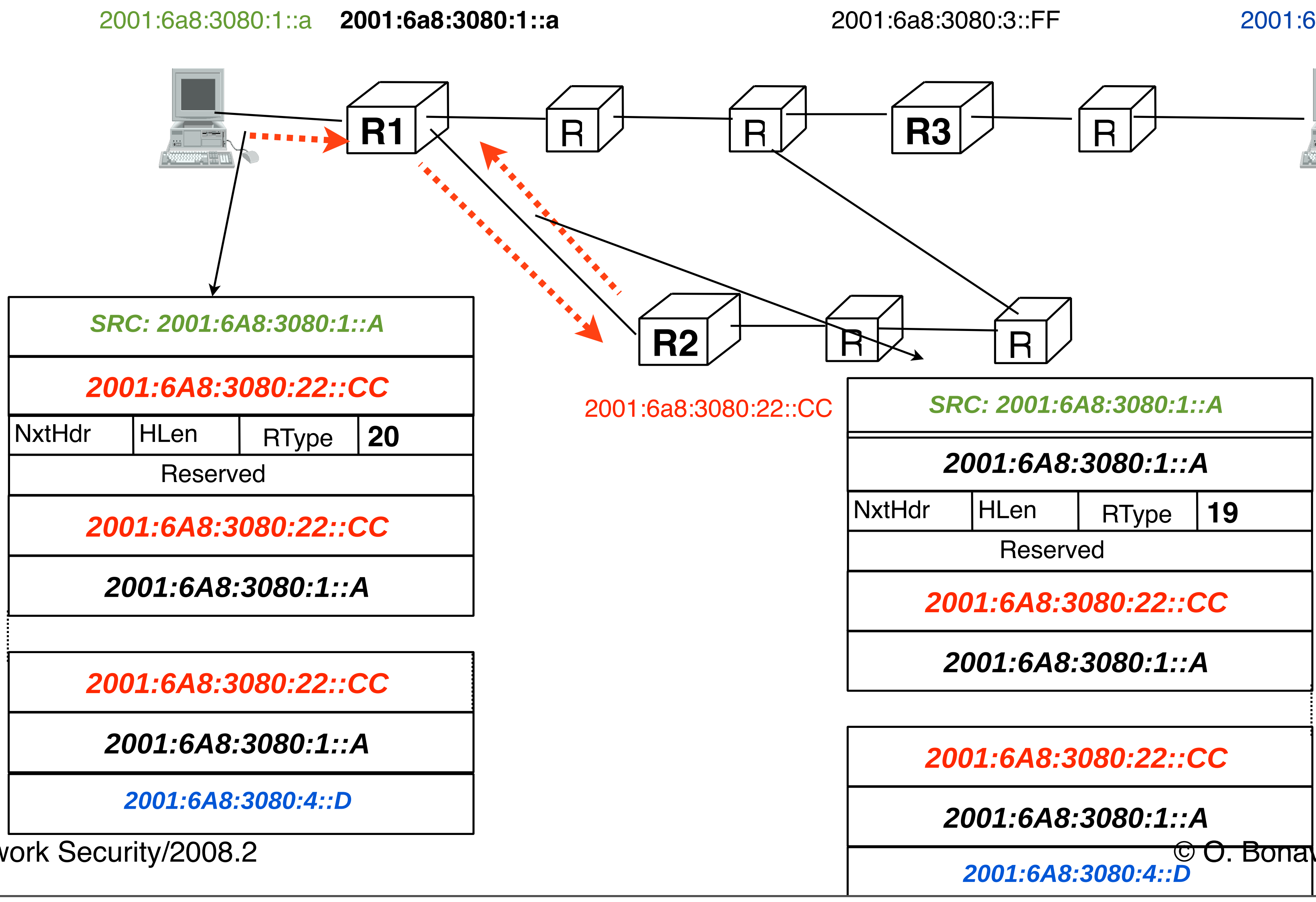


Problems with Type 0 RH



More serious problem with Type 0 RH

- Increases impact of DoS attacks



Hop-by-hop and destination option headers

- TLV format of these options

NxtHdr	HLen	Type	Len
Data (var. length)			

- Two leftmost bits
  - How to deal with unknown option ?
    - 00 ignore and continue processing
    - 01 silently discard packet
    - 10 discard packet and send ICMP parameter problem back to source
    - 11 discard packet and send ICMP parameter problem to source if destination isn't multicast
  - Third bit
    - Can option content be changed en-route
  - Five rightmost bits
    - Type assigned by IANA

The Len field encodes the size of the data field in bytes. Furthermore, special options have been defined to allow hosts using the options to pad the size of vairable length options to multiples of 64 bits.

Pad1 option (alignment requirement: none)

```
+-+-+-+-+-+-+-+-+
|       0       |
+-+-+-+-+-+-+-+-+
```

NOTE! the format of the Pad1 option is a special case -- it does not have length and value fields.

The Pad1 option is used to insert one octet of padding into the Options area of a header. If more than one octet of padding is required, the PadN option, described next, should be used, rather than multiple Pad1 options.

Deering & Hinden Standards Track [Page 10]

RFC 2460 IPv6 Specification December 1998

PadN option (alignment requirement: none)

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- - - - - - - - -
|       1       |  Opt Data Len |  Option Data
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- - - - - - - - -
```

The PadN option is used to insert two or more octets of padding into the Options area of a header. For N octets of padding, the Opt Data Len field contains the value N-2, and the Option Data consists of N-2 zero-valued octets.

IPv6 jumbograms

- IPv6 packet format only supports 64 KBytes packets
  - packet size is encoded in 16 bits field
- on most hosts throughput increases with packet size

- Hop-by-hop jumbogram option
  - Increases packet size to 32 bits
    - when used, packet size in IPv6 header should be set to zero

NxtHdr	HLen	C2	Len:4
Packet size			

C2 : 11 0 00020
11 -> ICMP must be sent if option is unrecognised
0 -> content of option does not change en-route

As of today, it is unclear whether the jumbogram option has been implemented in practice. Using it requires link layer technologies that are able to support frames larger than 64 KBytes.

The jumbogram option has been defined in

D. Borman, S. Deering, B. Hinden, IPv6 Jumbograms, RFC2675, August 1999

The Kame (http://www.kame.net) implementation on FreeBSD supports this option, but there is no link-layer that supports large frames.

Packet fragmentation

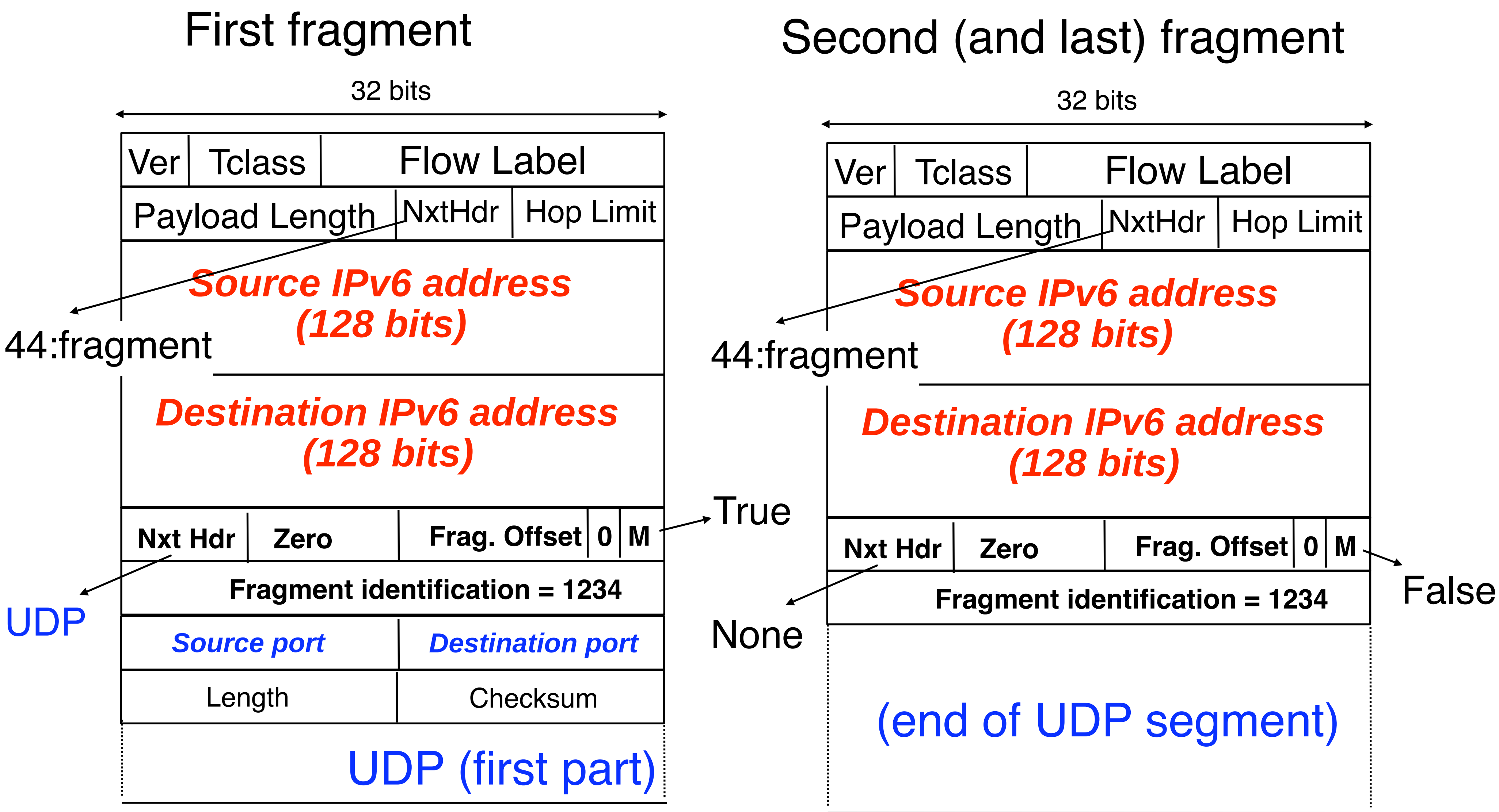
- IPv4 used packet fragmentation on routers
  - All hosts must handle 576+ bytes packets
  - experience showed fragmentation is costly for routers and difficult to implement in hardware
  - PathMTU discovery is now widely implemented

- IPv6
  - IPv6 requires that every link in the internet have an MTU of 1280 octets or more
    - otherwise link-specific fragmentation and reassembly must be provided at a layer below IPv6
  - Routers do not perform fragmentation
    - Only end hosts perform fragmentation and reassembly by using the fragmentation header
    - But PathMTU discovery should avoid fragmentation most of the time

Path MTU discovery is defined in

J. Mogul, S. Deering, Path MTU Discovery, RFC1191, 1996
and in
J. McCann, S. Deering, J. Mogul, Path MTU Discovery for IP version 6, RFC1981, 1996
for IPv6

A fragmented IPv6 packet



In IPv6, the fragment identification field is much larger than in IPv4. Furthermore, it is only used in packets that really need fragmentation. IPv6 header does not contain a fragmentation information for each unfragmented packet unlike IPv4.

IP version 6

- Outline
  - Motivations for IP version 6
  - IPv6 addressing architecture
  - IPv6 packets
  - ICMP v6

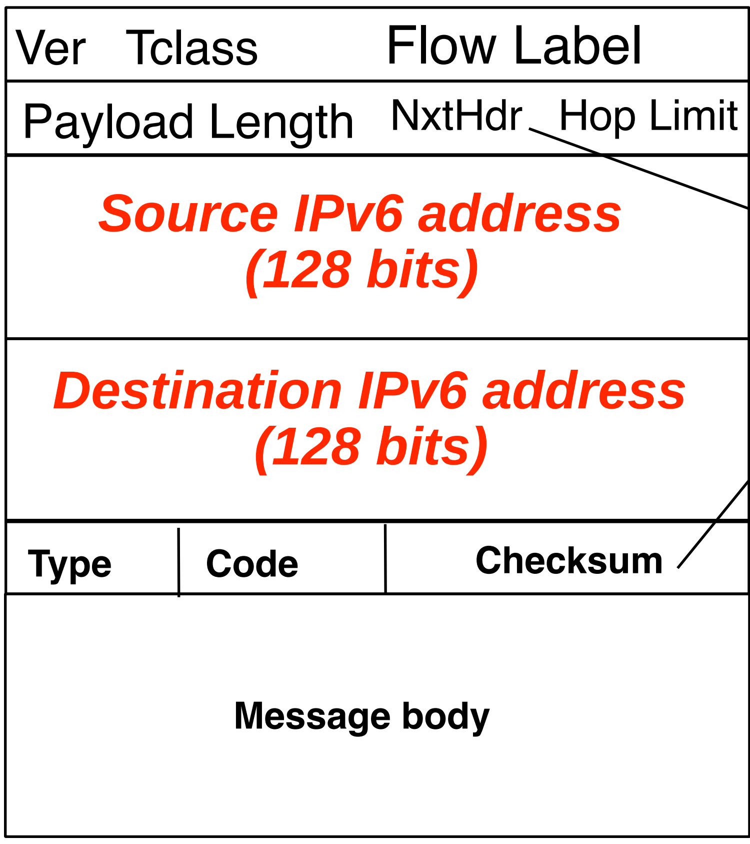
ICMPv6

- Provides the same functions as ICMPv4, IGMP and Address Resolution Protocol (ARP)

- Types of ICMPv6 messages
  - Destination unreachable
  - Packet too big
    - Used for PathMTU discovery
  - Time expired (Hop limit exhausted)
    - Traceroute v6
  - Echo request and echo reply
    - Pingv6
  - Multicast group membership
  - Router advertisments
  - Neighbor discovery
  - Autoconfiguration

ICMPv6 is defined in :
A. Conta, S. Deering, M. Gupta, Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification, RFC4443, March 2006

ICMPv6 packet format



58 for ICMPv6

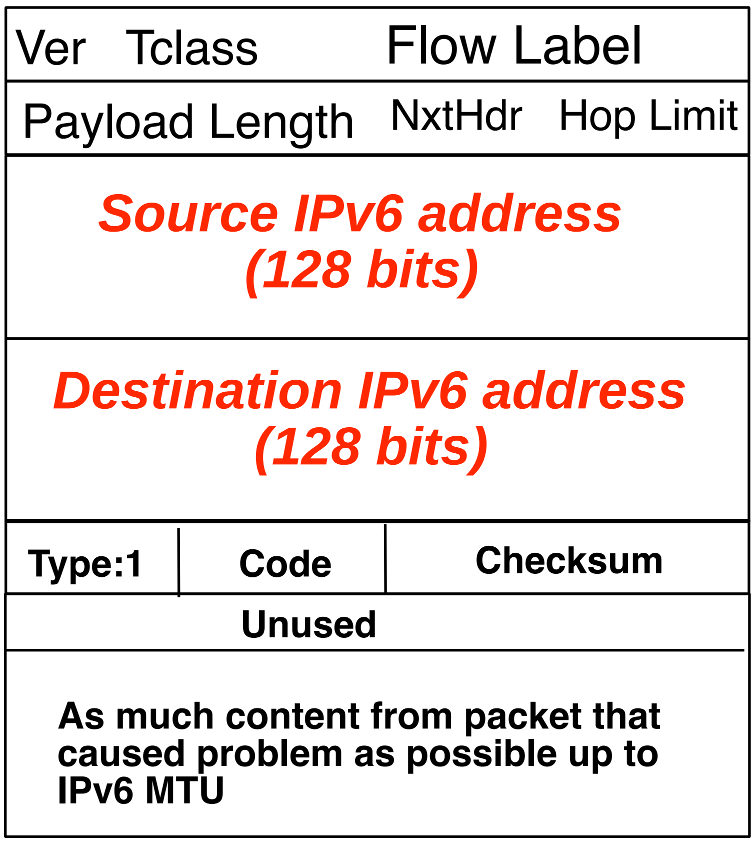
Covers ICMPv6 message and part of IPv6 header

- Type
- ICMPv6 error messages (0<type<127)
  - 1 Destination Unreachable
  - 3 Time Exceeded
  - 2 Packet Too Big
  - 4 Parameter Problem
  - 100 Private experimentation
  - 101 Private experimentation
  - 127 Reserved for expansion
- ICMPv6 informational messages:
  - 128 Echo Request
  - 129 Echo Reply
  - 200 Private experimentation
  - 201 Private experimentation
  - 255 Reserved for expansion

of ICMPv6 informational

ICMPv6 uses a next header value of 58 inside IPv6 packets

ICMPv6 destination unreachable



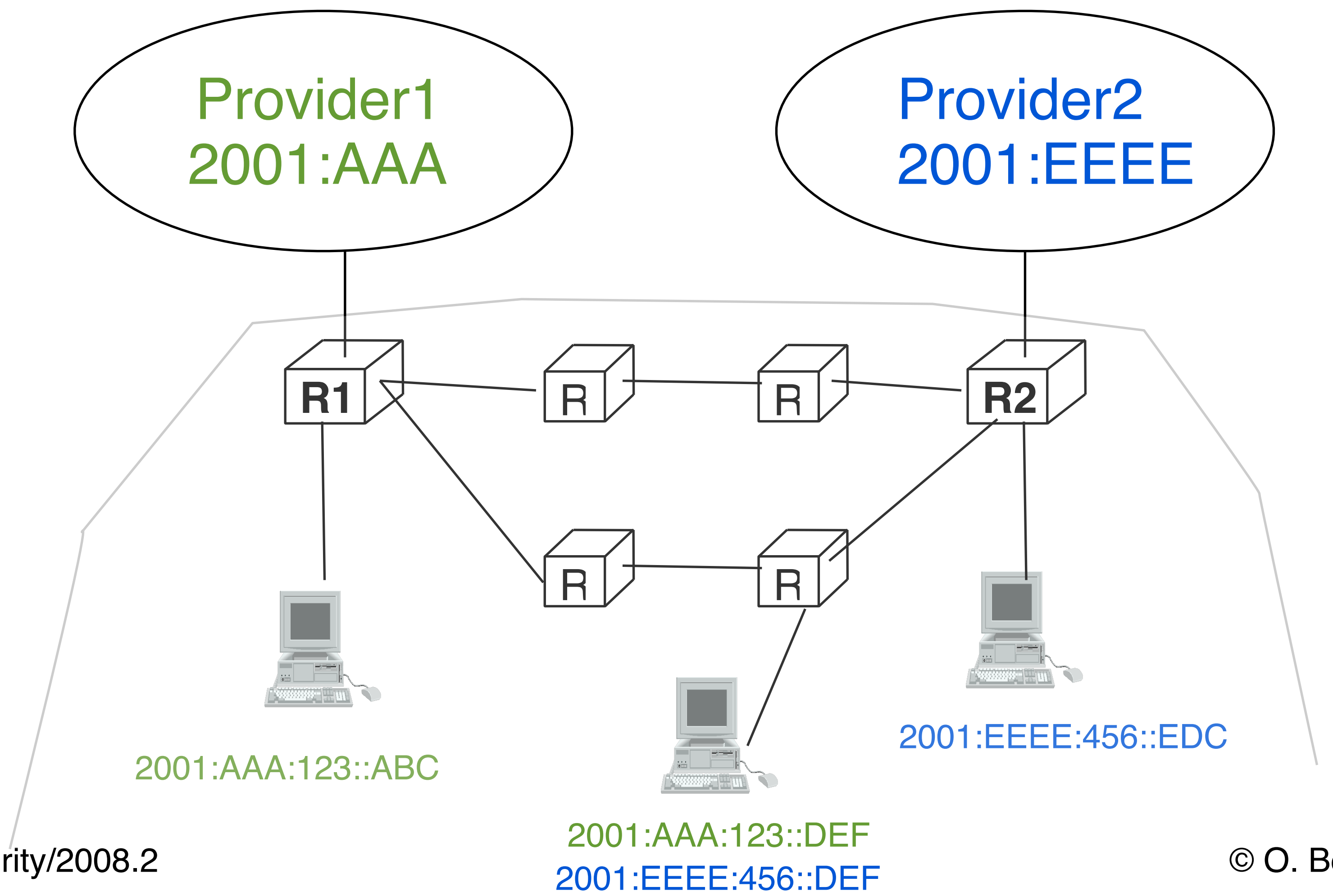
- Code
  - 0 - No route to destination
  - 1 - Communication with destination administratively prohibited
  - 2 - Beyond scope of source address
  - 3 - Address unreachable
  - 4 - Port unreachable
  - 5 - Source address failed ingress/egress policy
  - 6 - Reject route to destination

The Unused field is used to align the content of the ICMPv6 message to a 64 bits boundary.

Note that for security reasons, it is recommended that implementations should allow sending of ICMP destination unreachable messages to be disabled, preferably on a per-interface basis.

Ingress and egress policies

- For security reasons, a provider should only accept packets from sources belonging to allocated prefixes

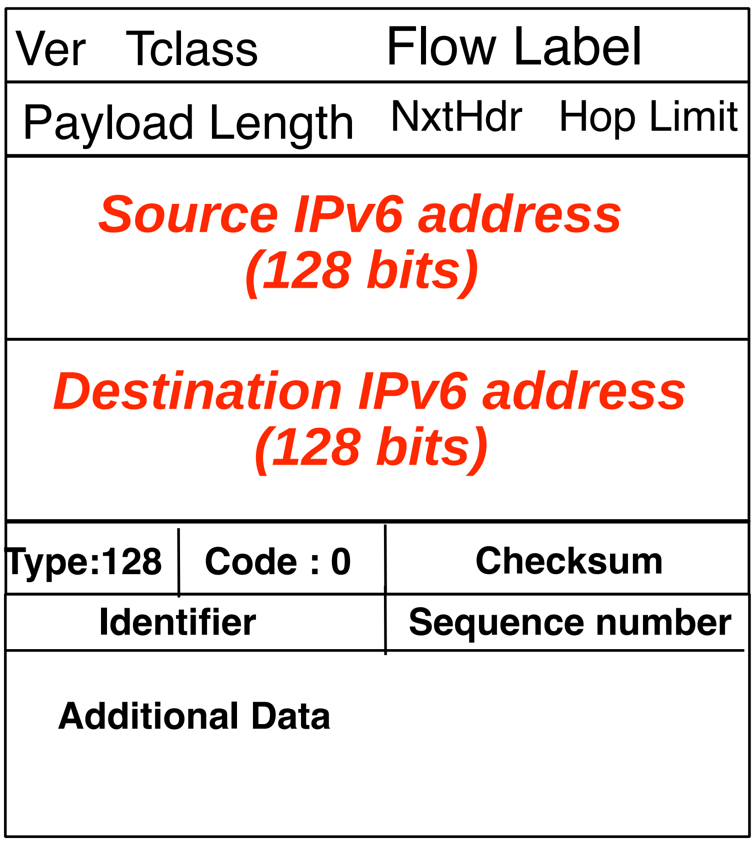


These policies are described in

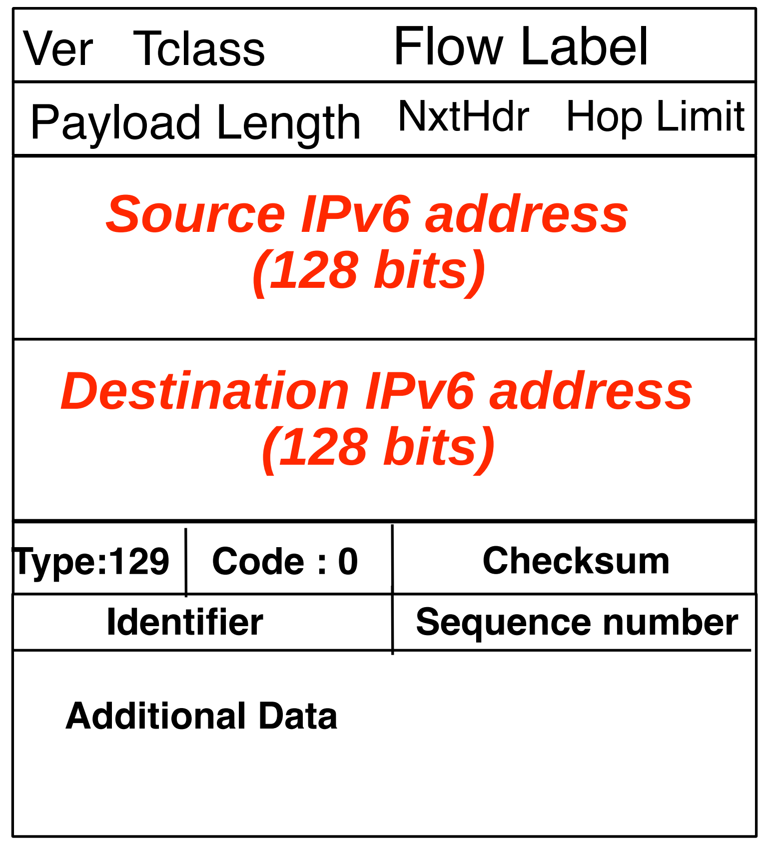
F. Baker, P. Savola, Ingress Filtering for Multihomed Networks, RFC3704, March 2004

ICMPv6
echo request and reply

Echo request



Echo reply

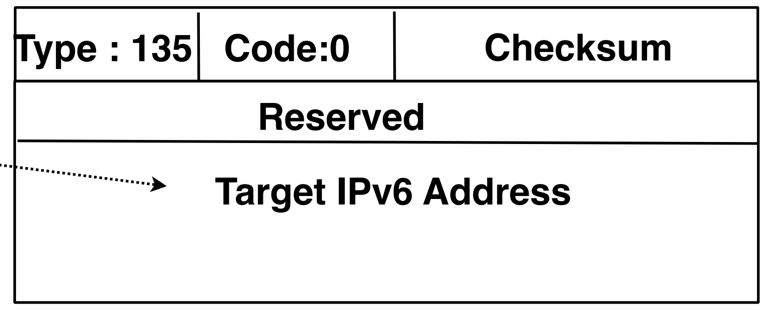


- Identifier and sequence number
  - chosen by source to aid in correlating reply with request
  - copied by destination when generating echo reply

ICMPv6 Neighbour Discovery

- Replacement for IPv4's ARP
- Neighbour solicitation

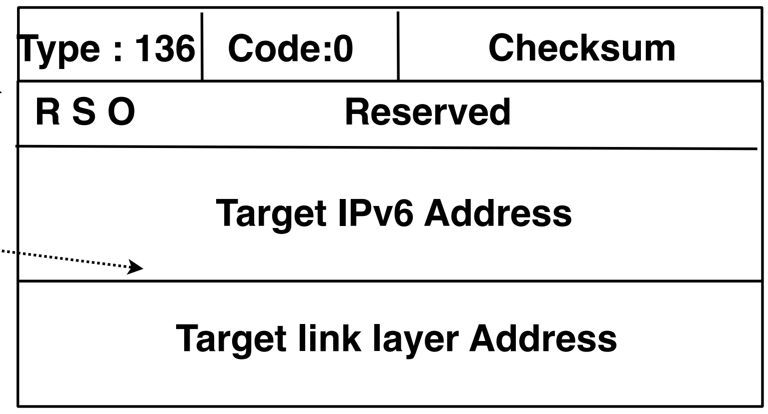
The IPv6 address for which the link-layer (e.g. Ethernet) address is needed.
May also contain an optional field with the link-layer (e.g. Ethernet) address of the sender.



- Neighbour advertisement

R : true if node is a router
S : true if answers to a neighbour solicitation

The IPv6 and link-layer addresses



The ICMPv6 neighbour discovery messages are sent with HopLimit=255

The role of the R, S and O flags is described as follows in RFC4861

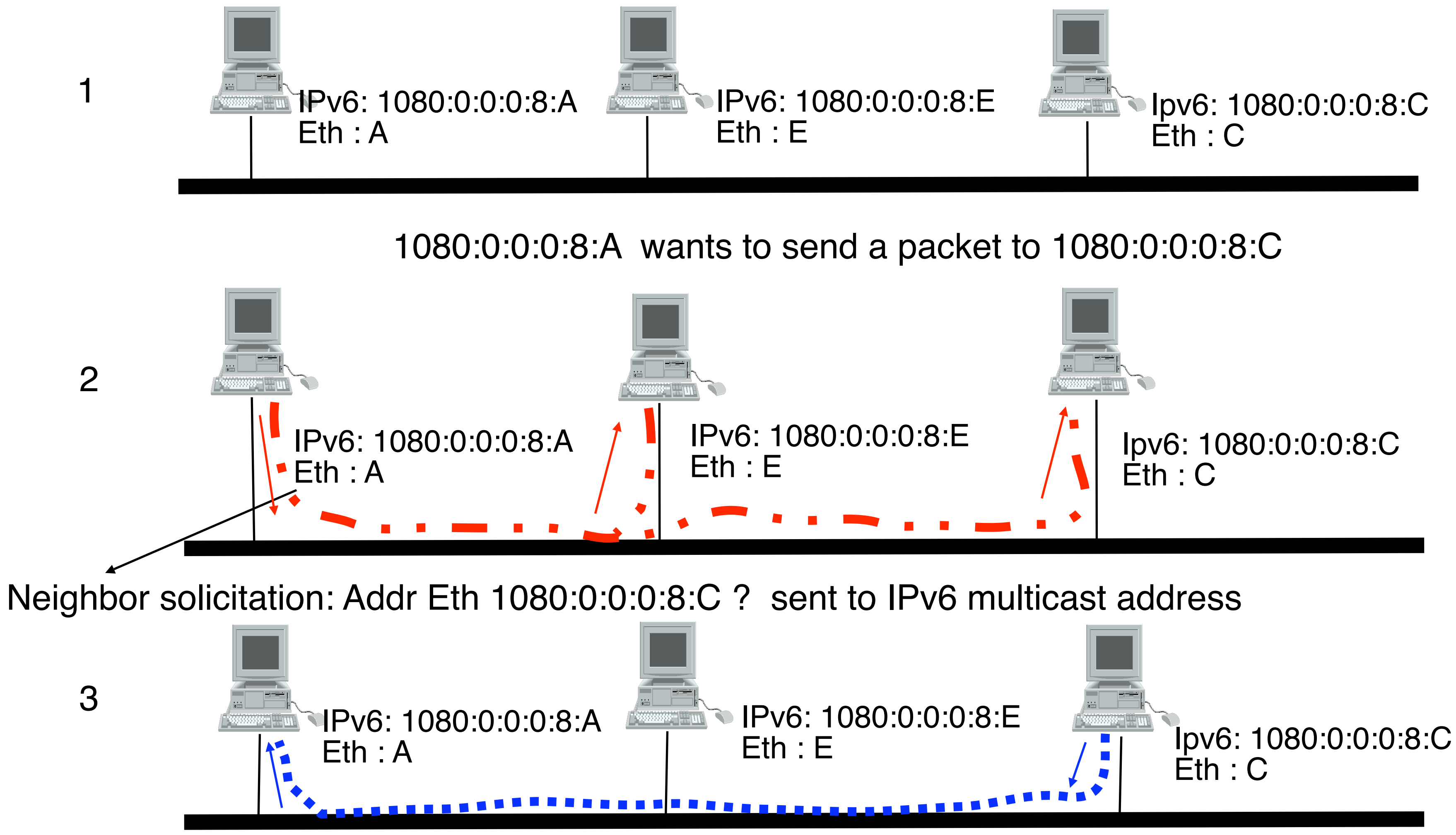
R Router flag. When set, the R-bit indicates that the sender is a router. The R-bit is used by Neighbor Unreachability Detection to detect a router that changes to a host.

S Solicited flag. When set, the S-bit indicates that the advertisement was sent in response to a Neighbor Solicitation from the Destination address. The S-bit is used as a reachability confirmation for Neighbor Unreachability Detection. It MUST NOT be set in multicast advertisements or in unsolicited unicast advertisements.

O Override flag. When set, the O-bit indicates that the advertisement should override an existing cache entry and update the cached link-layer address. When it is not set the advertisement will not update a cached link-layer address though it will update an existing Neighbor Cache entry for which no link-layer address is known. It SHOULD NOT be set in solicited advertisements for anycast addresses and in solicited proxy advertisements. It SHOULD be set in other solicited advertisements and in unsolicited advertisements.

IPv6 over Ethernet

- Neighbour discovery / address resolution



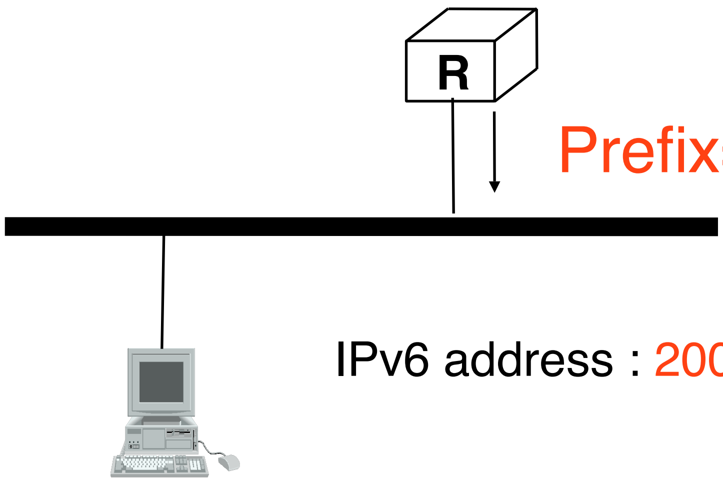
Neighbor advertisement: 1080:0:0:0:8:C is reachable via Ethernet Add : C

The transmission of IPv6 packets over Ethernet is defined in :
M. Crawford, Transmission of IPv6 Packets over Ethernet Networks, RFC2464, December 1998

Note that in contrast with ARP used by IPv4, ICMPv6 neighbour solicitation messages are sent to a multicast ethernet address and not to the broadcast ethernet address. This implies that only the IPv6 enabled hosts on the LAN will receive the ICMPv6 message.

IPv6 autoconfiguration

- How can a node obtain its IPv6 address ?
  - Manual configuration
  - From a server by using DHCPv6 as in IPv4
- Automatically
  - Router advertises prefix on LAN by sending ICMPv6 messages to “all IPv6 hosts” multicast address
  - Hosts build their address by concatenating the prefix with their MAC Address converted in 64 bits format



Prefix= 2001:6a8:3080:1/64

IPv6 address : 2001:6a8:3080:1:M_{64}(800:200C:417A)

Ethernet : 0800:200C:417A

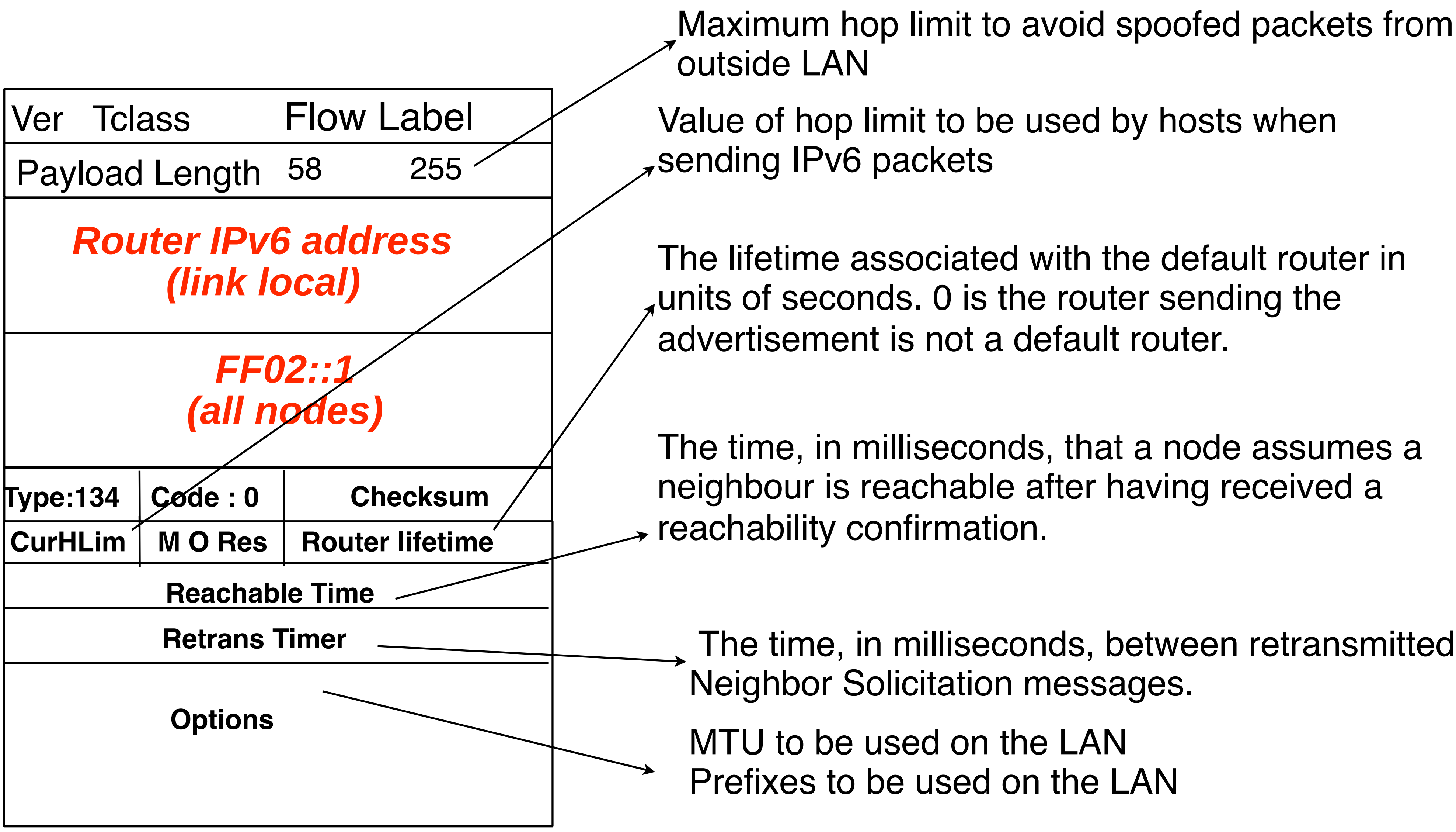
M_{64}(800:200C:417A) is a function that converts a 48 bits MAC address into a 64 bits Interface Identifier. This function is defined in :

R. Hinden, S. Deering, IP Version 6 Addressing Architecture, RFC4291, February 2006

The IPv6 autoconfiguration is defined in :

S. Thomson, T. Narten, T. Jinmei, IPv6 Stateless Address Autoconfiguration, RFC4862, Sept. 2007

Router advertisements



When the M bit is set to true, this indicates that IPv6 addresses should be obtained from DHCPv6

When the O bit is set to true, this indicates that the hosts can obtain additional information (e.g. address of DNS resolver) from DHCPv6

The router advertisements messages can also be sent in unicast in response to solicitations from hosts. A host can obtain a router advertisement by sending a router solicitation which is an ICMPv6 message containing only the router solicitation message (type 133).

Router advertisements options

- Format of the options

Type	Length	Options
Options (cont.)		

- MTU option

Type : 5	Length:1	Reserved
MTU		

- Prefix option

Type : 3	Length:4	PreLen	L A Res.
Valid Lifetime			
Preferred Lifetime			
Reserved2			
IPv6 prefix			

Number of bits in IPv6 prefix that identify subnet

The validity period of the prefix in seconds

The duration in seconds that addresses generated from the prefix via stateless address autoconfiguration remain preferred.

The two L and A bits are defined as follows :

L 1-bit on-link flag. When set, indicates that this prefix can be used for on-link determination. When not set the advertisement makes no statement about on-link or off-link properties of the prefix. In other words, if the L flag is not set a host MUST NOT conclude that an address derived from the prefix is off-link. That is, it MUST NOT update a previous indication that the address is on-link.

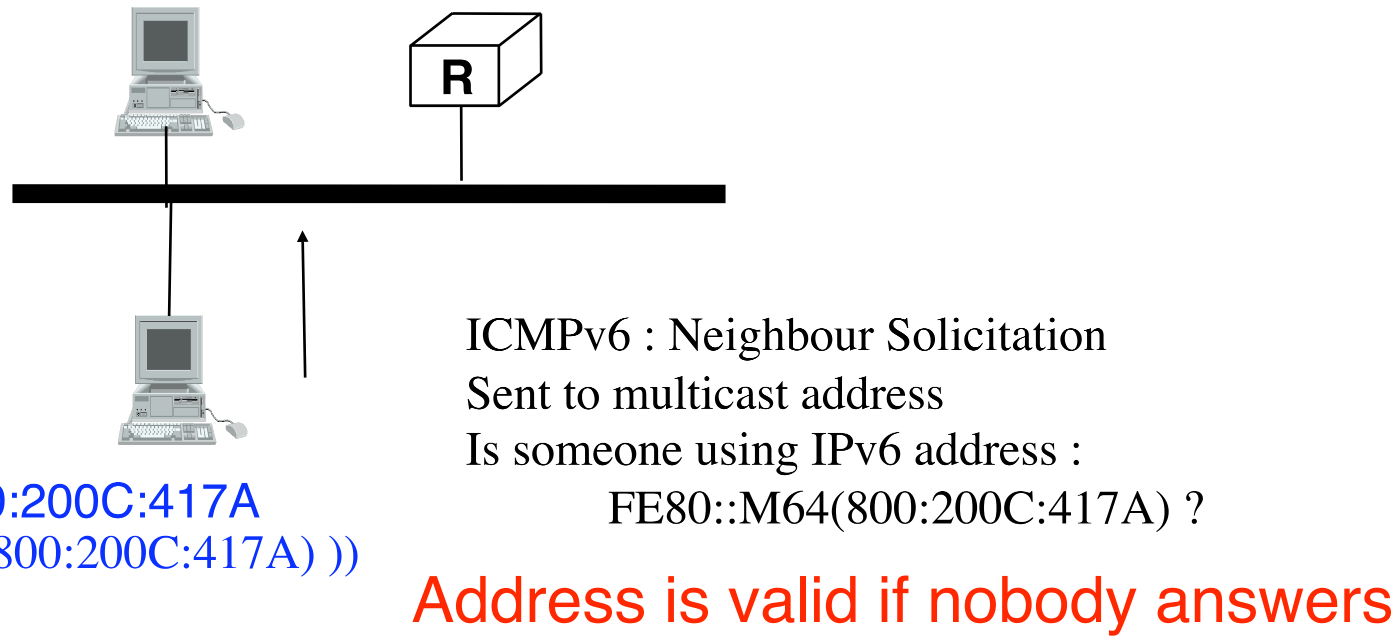
A 1-bit autonomous address-configuration flag. When set indicates that this prefix can be used for stateless address configuration.

Other options have been defined for the router advertisements. For example, the RDNSS option defined in
J. Jeong, S. Park, L. Beloeil, S. Madanapalli, IPv6 Router Advertisement Option for DNS Configuration, RFC 5006, Sept. 2007

allows a router to advertise the IPv6 address of the DNS resolver to be used by hosts on the LAN.

IPv6 autoconfiguration (2)

- What happens when an endsystem boots ?
  - It knows nothing about its current network
    - but needs an IPv6 address to send ICMPv6 messages

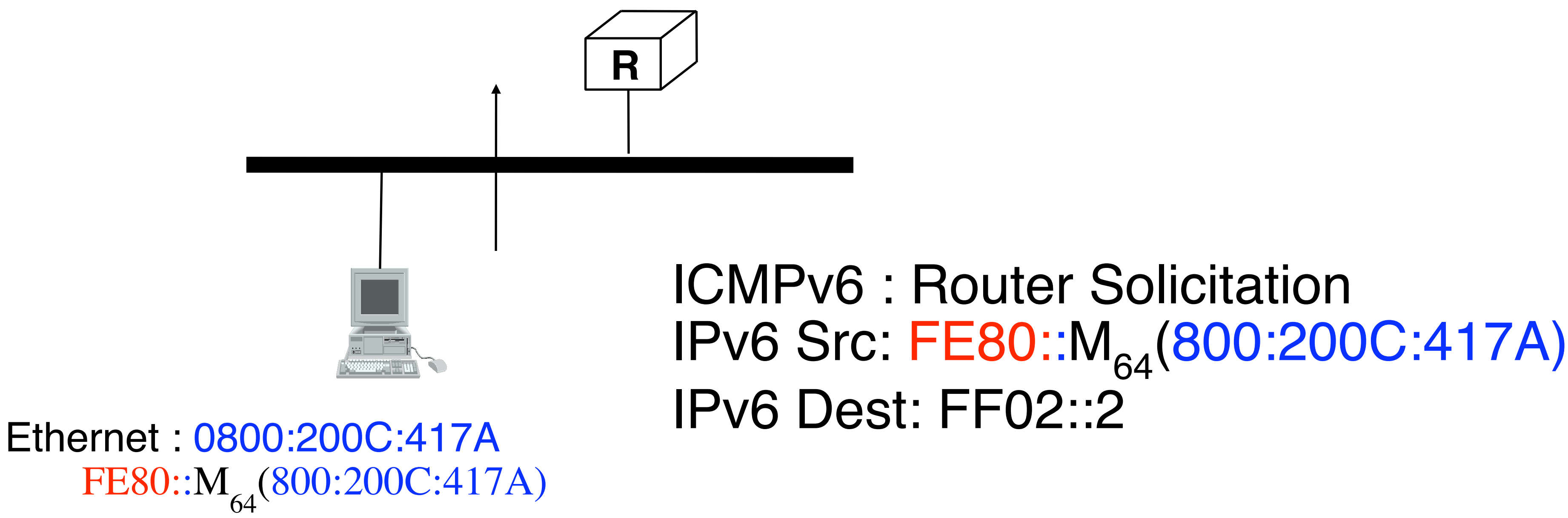


- Use Link-local IPv6 address (FE80::/64)
  - Each host, when it boots, has a link-local IPv6 address
  - But another node might have chosen the same address !

This utilisation of ICMPv6 Neighbour solicitation is called Duplicate Address Detection. It is used everytime a host obtains a new IPv6 address and is required to ensure that a host
is not using the same IPv6 address as another host on the same LAN.

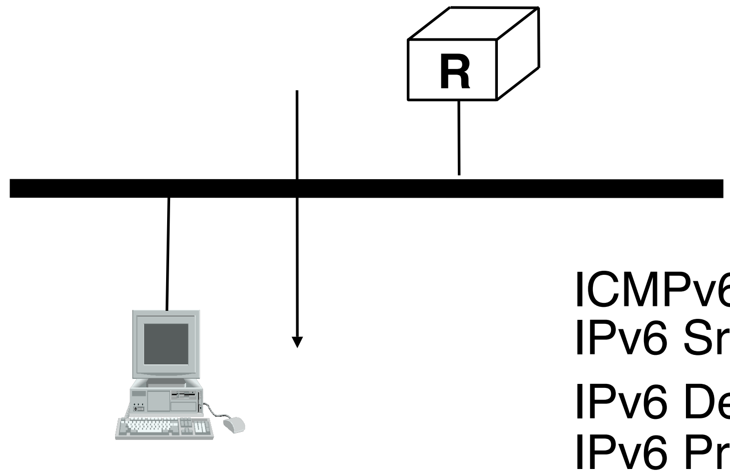
IPv6 autoconfiguration (2)

- How to obtain the IPv6 prefix of the subnet ?
  - Wait for router advertisements (e.g. 30 seconds)
  - Solicit router advertisement



IPv6 autoconfiguration (3)

- Router will re-advertise prefix



ICMPv6 : Router Advertisement
IPv6 Src: FE80::M_{64}(EthernetR)
IPv6 Dest: FF02::1
IPv6 Prefix = 2001:6a8:1100::/48
Prefix lifetime

Ethernet : 0800:200C:417A
FE80::M_{64}(800:200C:417A)

- IPv6 addresses can be allocated for limited lifetime
  - This allows IPv6 to easily support renumbering

IPv6 is supposed to easily support renumbering and IPv6 router advertisements are one of the ways to perform this renumbering by allowing hosts t update their IPv6 addresses upon reception of new router advertisement messages. However, in practice renumbering an IPv6 network is not easily because IPv6 addresses are manually encoded in too many configuration files, see e.g. :

F. Baker, E. Lear, R. Droms, Procedures for Renumbering an IPv6 Network without a Flag Day, RFC4192, 2005

Privacy issues with IPv6 address autoconfiguration

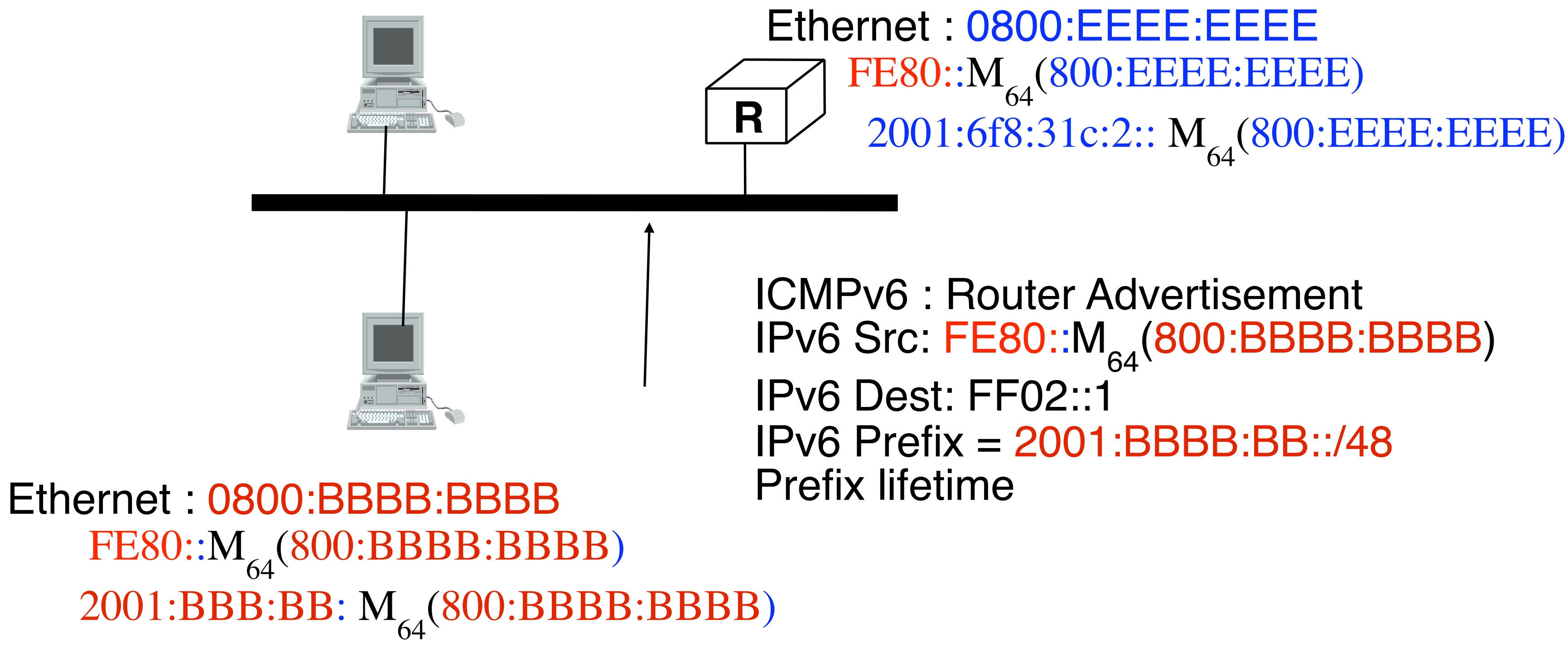
- Issue
  - Autoconfigured IPv6 addresses contain the MAC address of the hosts
    - MAC addresses are fixed and unique
    - A laptop/user could be identified by tracking the lower 64 bits of its IPv6 addresses
- How to maintain privacy with IPv6 ?
  - Use DHCPv6 and configure server to never reallocate the same IPv6 address
  - Allow hosts to use random host ids in lower 64 bits of their IPv6 address
    - algorithms have been implemented to generate such random host ids on nodes with and without stable storage

This extension to support privacy-aware IPv6 addresses is defined in

T. Narten, R. Draves, S. Krishnan, Privacy Extensions for Stateless Address Autoconfiguration in IPv6, RFC4941, Sept. 2007

Security risks

- What happens if an attacker sends fake router advertisements on LAN ?



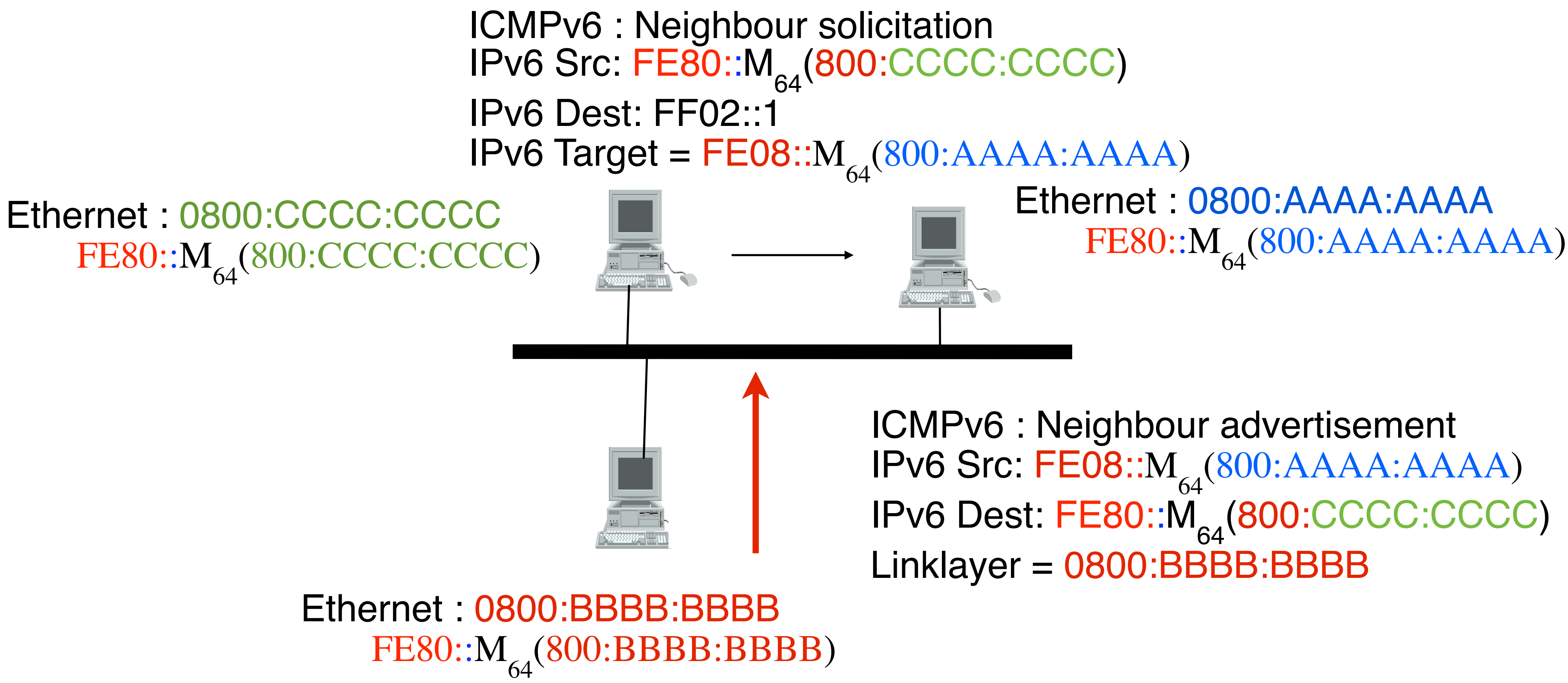
Risk of man-in-the-middle attack, other hosts could use the attacker as their default router

A discussion of the security issues with Neighbour discovery may be found in

P. Nikander, J. Kempf, E. Nordmark, IPv6 Neighbor Discovery (ND) Trust Models and Threats, RFC3756, May 2004

Security risks (2)

- What happens if an attacker sends fake ICMPv6 neighbour advertisements ?



Securing ICMPv6

- Principle of the solution
  - A host that replies to an ICMPv6 neighbour solicitation should be able to prove that it owns the corresponding IPv6 address
  - A router that sends router advertisements should be able to prove that it is authorised to serve as a router using the advertised prefixes
- Issues
  - How to exchange theses proofs and authorisations ?
  - Is IPSec a solution ?

First solution : certificates

- Principle
  - Each router has a public/private keypair
  - A certificate is generated for each router to confirm :
    - that the keypair belongs to the router
    - that the owner of the keypair is a valid router
  - Certificate must be anchored on an authority that is trusted by both routers and hosts
  - ICMPv6 router advertisement messages are signed by the router
- Protocol issues
  - Need to extend ICMPv6 to support signatures and certificates

Additional information about the utilisation of X.509 certificates to represent IP prefixes and AS resources, see :

Lynn, C., Kent, S. and K. Seo, "X.509 Extensions for IP Addresses and AS Identifiers", RFC 3779, June 2004.

The development of these certificates is being performed within the SIDR working group of the IETF.

Cryptographically Generated Addresses

- Placing certificates on all hosts is too difficult
- We usually don't need to prove that a host is a host

- Can we verify the validity of signed messages without relying on a PKI ?

- Principle of the solution
  - Assume that IPv6 addresses are variable-length
  - Generate IPv6 addresses as follows

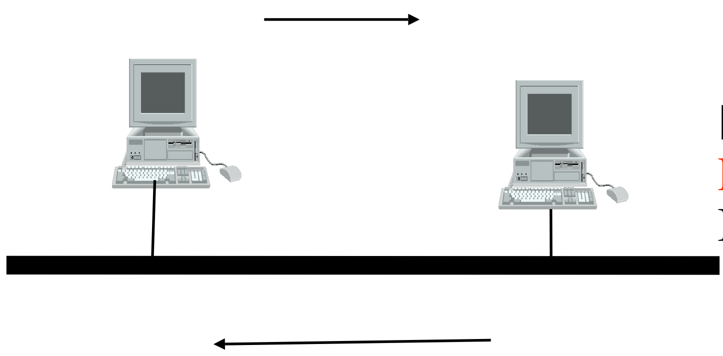
Global prefix + subnet id (64bits)	Host's public key

  - Use private key to sign ICMPv6 neighbour advertisement messages

Principles of Secure Neighbour Discovery

ICMPv6 : Neighbour solicitation
IPv6 Src: FE80:: KeyC
IPv6 Dest: FF02::1
IPv6 Target = FE80::KeyA
Nonce=1234
Timestamp : April14,2008, 10.00:01

Ethernet : 0800:CCCC:CCCC
Public key : KeyC
IPv6 : FE80::KeyC



Ethernet : 0800:AAAA:AAAA
Public key : KeyA
IPv6 : FE80::KeyA

ICMPv6 : Neighbour Advertisement
IPv6 Src: FE80::KeyA
IPv6 Dest: FE80::KeyC
IPv6 Target = FE80::KeyA
Nonce=1234
CGA Parameter : KeyA...
Timestamp : April14,2008, 10.00:07
Signature : Message signed with KeyA

Cryptographically Generated Addresses

- IPv6 addresses have a fixed size
  - Unfortunately, only 62 bits are available in host id
    - A 62 bits RSA public-key is not secure
- Solution
  - To secure a binding between a MAC address and an IPv6 address, each host
    - generates its ($public_{key}$,$private_{key}$) key pair
    - uses a special HostId = $Hash_{62}(public_{key})$
    - Signs the Neighbour advertisement by using its $private_{key}$

The utilisation of a 62 bits hash instead of a 64 bits hash is necessary because some bits of the host id part of the IPv6 address are reserved. When using CGAs, the two high order bits of the hostid must be set to 0 to indicate that this host id is not globally unique

Cryptographically Generated Addresses (2)

- Issue with CGA
  - A 62 bits hash is not very secure
    - an attacker could use brute-force to find a public-key whose hash is equal to a given value

- Improving CGA security beyond 62 bits
  - Increases the difficulty of computing $Hash_{62}(public_{key})$
  - Define security parameter, Sec=0,1,2,3
    - Encode Sec in 2 high order bits of HostId
    - If Sec=0, then HostId = $Hash_{62}(Random \mid public_{key})$
    - If Sec=1, then
      Find Random : $High_{20}(Hash_{80}(Random \mid public_{key}))=0$
      HostId = $Low_{60}(Hash_{80}(Random \mid public_{key}))$
    - ...

This is a simplified description of the computation of a cryptographically generated address. For more details, see :

J. Arkko et al. Securing IPv6 Neighbor and Router Discovery, WiSe 02, September 2002

Lynn, C., Kent, S. and K. Seo, "X.509 Extensions for IP Addresses and AS Identifiers", RFC 3779, June 2004.

Aura, T., "Cryptographically Generated Addresses (CGA)", RFC3972, March 2005.

Cryptographically Generated Addresses (3)

- Issues with CGA
  - The HostId should not only depend on public key
- Solution
  - CGA depends on several parameters
    - Modifier
      - 16 octets random value
    - Subnet prefix
      - 8 octets
    - Collision Count
      - Incremented each time a duplicate address is found
    - Public key

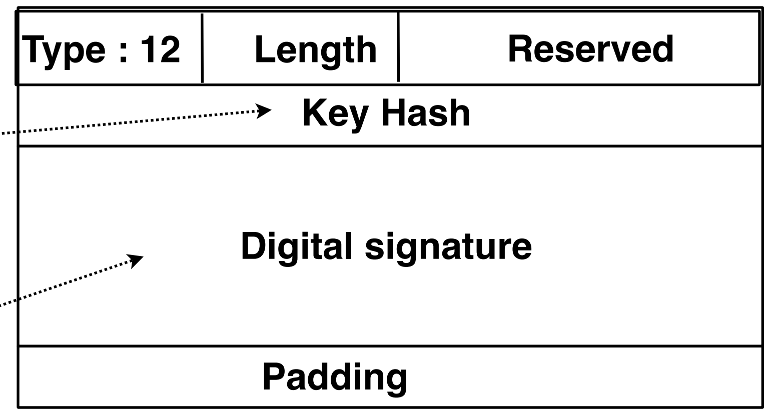
The structure described above will be send by the endsystem in the neighbor advertisement and will be used by the recipient of the message to check the validity of the signature.

The utilization of CGA by the Neighbor Discovery protocol for IPv6 is defined in :

J. Arkko, J. Kempf, B. Sommerfeld, B. Zill, P. Nikander, Secure Neighbor Discovery (SEND), Internet draft, draft-ietf-send-ndopt-06.txt, July 2004, work in progress

Extensions to ICMPv6

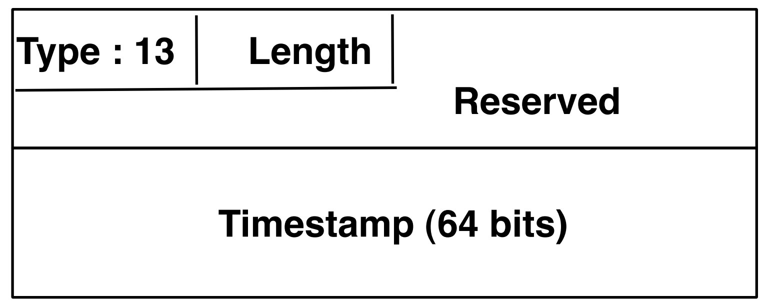
- Signature option



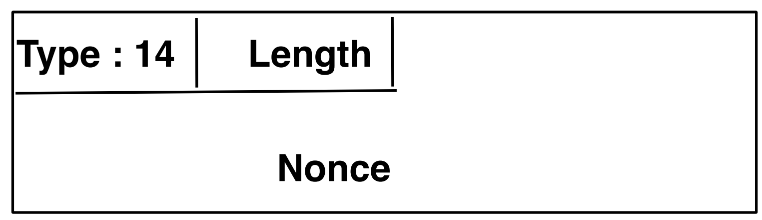
SHA-1 hash (most significant 128 bits)
of the public key used to compute signature.

The signature is computed over the following information :
- random message tag
- 128 bits source address of IPv6 header
- 128 bits destination address of IPv6 header
- Type, Code and Checksum of ICMPv6 header
- NDP message header and options

- Timestamp option
  - used to avoid replay attacks



- Nonce option



See Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

The random message tag is (0x086F CA5E 10B2 00C9 9C8C E001 6427 7C08.) This value was chosen at random by the editor of the above document.

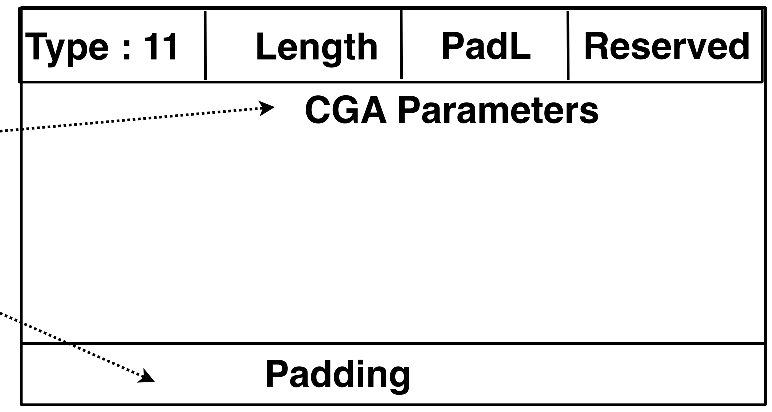
A nonce option is also defined. This option is used to secure the replies sent by routers to neighbour solicitations.

Extensions to ICMPv6 (2)

- CGA option

Parameters used to compute the CGA address

Padding to ensure that CGA option is n*8 bytes



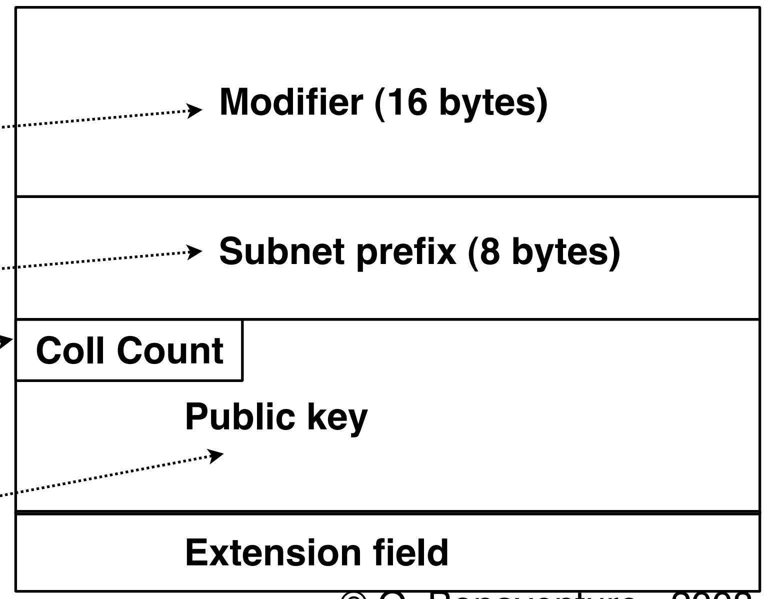
- CGA Parameters

Random value, used to add randomness in the generation of the CGA to improve privacy

The subnet prefix where the address resides

Number of collision in CGA generation

RSA public key, at least 384 bits

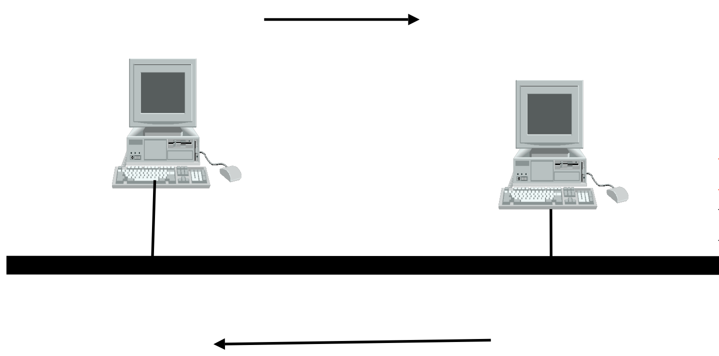


T. Aura, Cryptographically Generated Addresses (CGA) , RFC3972, March 2005

Secure Neighbour Discovery

ICMPv6 : Neighbour solicitation
IPv6 Src: FE80:: Hash(KeyC)
IPv6 Dest: FF02::1
IPv6 Target = FE80::Hash(KeyA)
Nonce=1234
Timestamp : April14,2008, 10.12:01

Ethernet : 0800:CCCC:CCCC
Public key : KeyC
IPv6 : FE80::KeyC



Ethernet : 0800:AAAA:AAAA
Public key : KeyA
IPv6 : FE80::KeyA

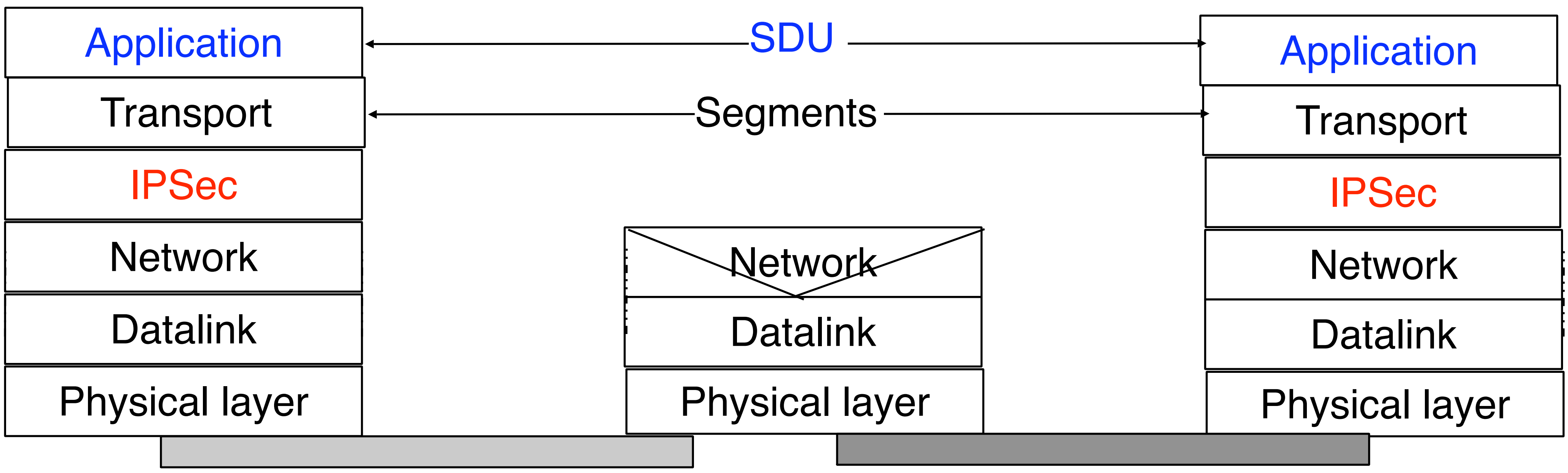
ICMPv6 : Neighbour Advertisement
IPv6 Src: FE80::Hash(KeyA)
IPv6 Dest: FE80::Hash(KeyC)
IPv6 Target = FE80::KeyA
Nonce=1234
CGA Parameter : KeyA...
Timestamp : April14,2008, 10.12:07
Signature : Message signed with KeyA

Internet and Network security

- Crypto building blocks
- Application-layer security
- Transport-layer security
- Network-layer security
  - IPv4
  - IPv6
  - IPSec
  - Routing security

IPSec

- Principle
  - Protect the IP layer by adding encryption and authentication on a per IP packet basis
    - IPv4 and IPv6



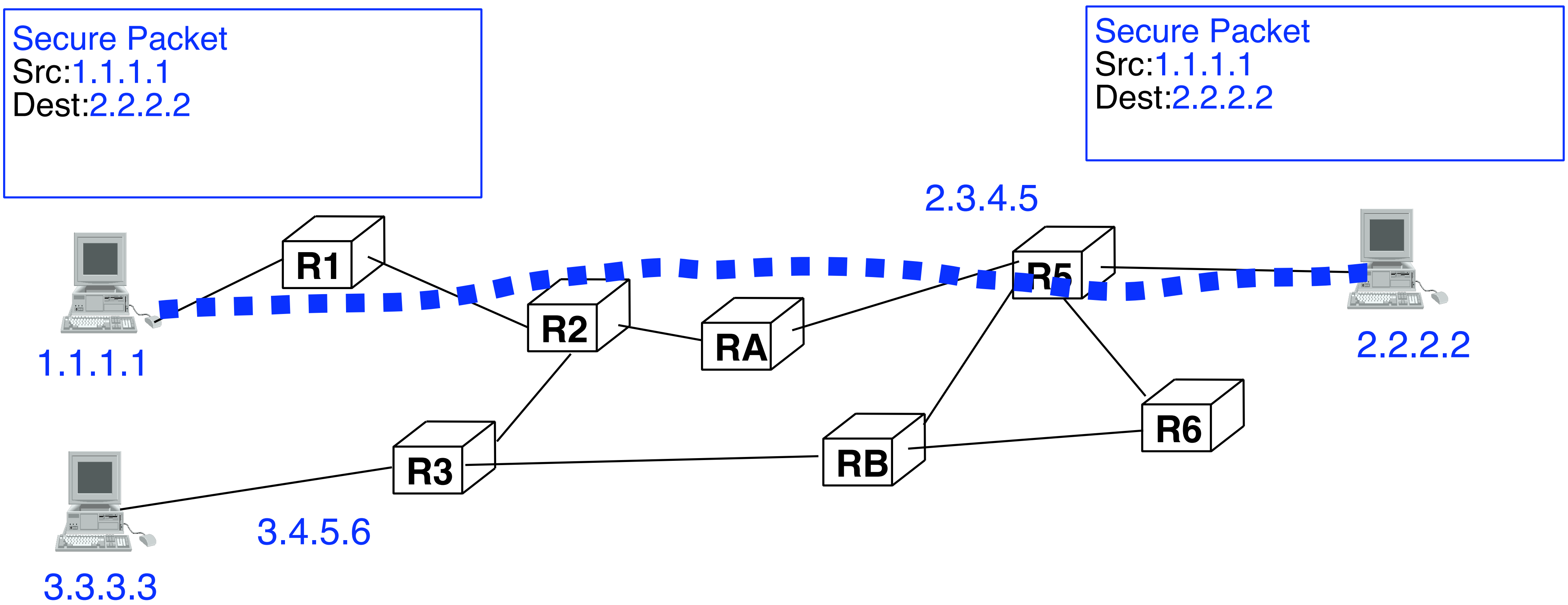
Descriptions of IPSec may be found in :

N. Doraswany, D. Harkins, IPSec : The new security standard for the Internet, Intranets and Virtual Private Networks, Prentice Hall, 1999

S. Frankel, Demystifying the IPsec Puzzle, Artech House, 2001

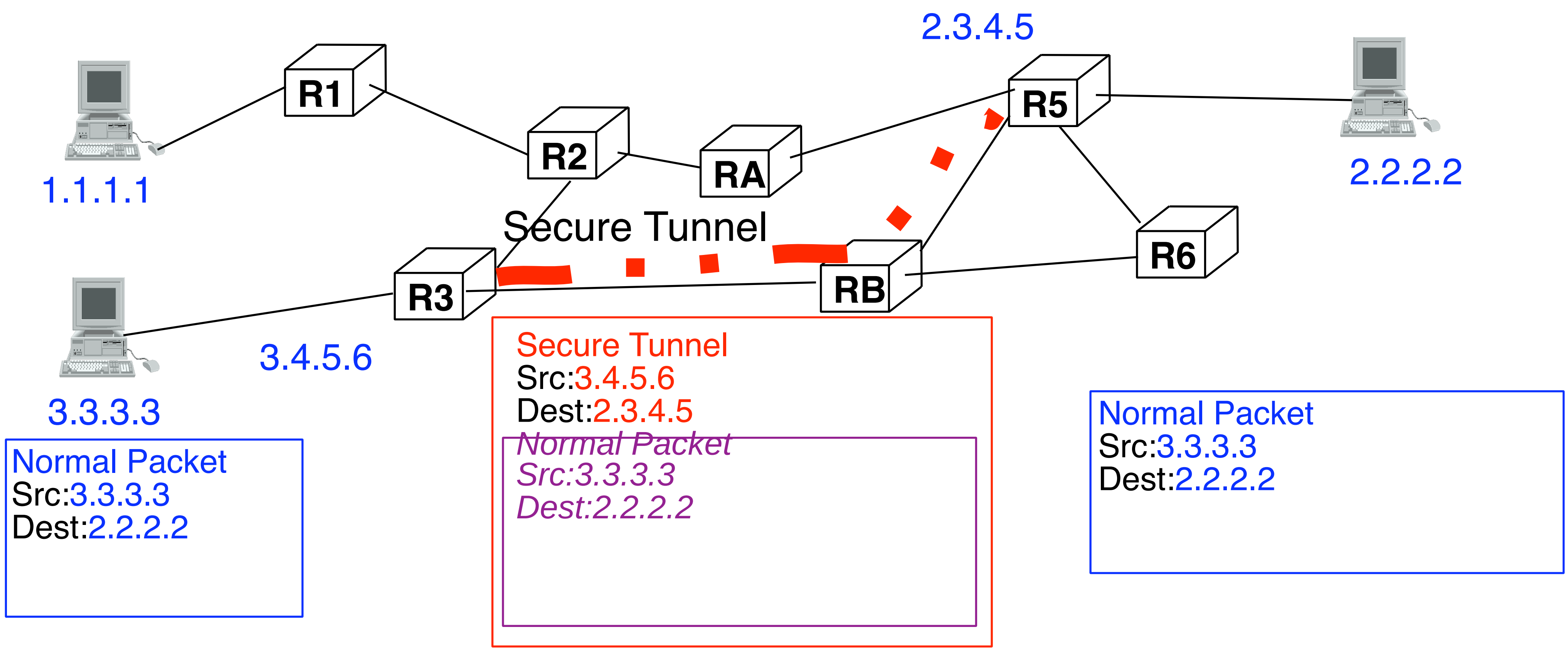
Modes of operation of IPSec

- Transport Mode
  - End-to-end protection of IP packets



Modes of operation of IPSec (2)

- Tunnel Mode
  - Router-to-router protection of IP packets



Behaviour of an IPSec node

- How does a node decide which packets need to be encrypted/authenticated ?
  - Each node contains a Security Policy Database defining which packets needs to be secured
    - SPD decision is based on any packet header (e.g. Destination address, source address, ...)
  - Example
    - Secure all telnet traffic
    - Secure packets sent to bank offices, but not to Internet
    - Secure UDP
    - Secure TCP but not SSL
    - ...

The security Policy Database and Security Associations are defined in RFC2401

The IPSEC architecture is defined in :

S. Kent, K. Seo, Security Architecture for the Internet Protocol, RFC4301, December 2005

Behaviour of an IPSec node (2)

- How to send secure packets ?
  - Nodes willing to exchange packets securely must establish a Security Association (SA)
    - Internet Key Exchange protocol used for authentication and key exchange during SA establishment
    - each communication node maintains state for each SA inside its Security Association Database
    - Several SAs may be established between two nodes

Perfect Forward Secrecy

- One of the objectives of IPSec
  - A protocol provides Perfect Forward Secrecy if it is impossible for an eavesdropper to decrypt a conversation between Alice and Bob by :
    - capturing all packets including key exchange
    - breaking after the conversation into Alice's and Bob's computers to steal their secrets (e.g. Public keys)

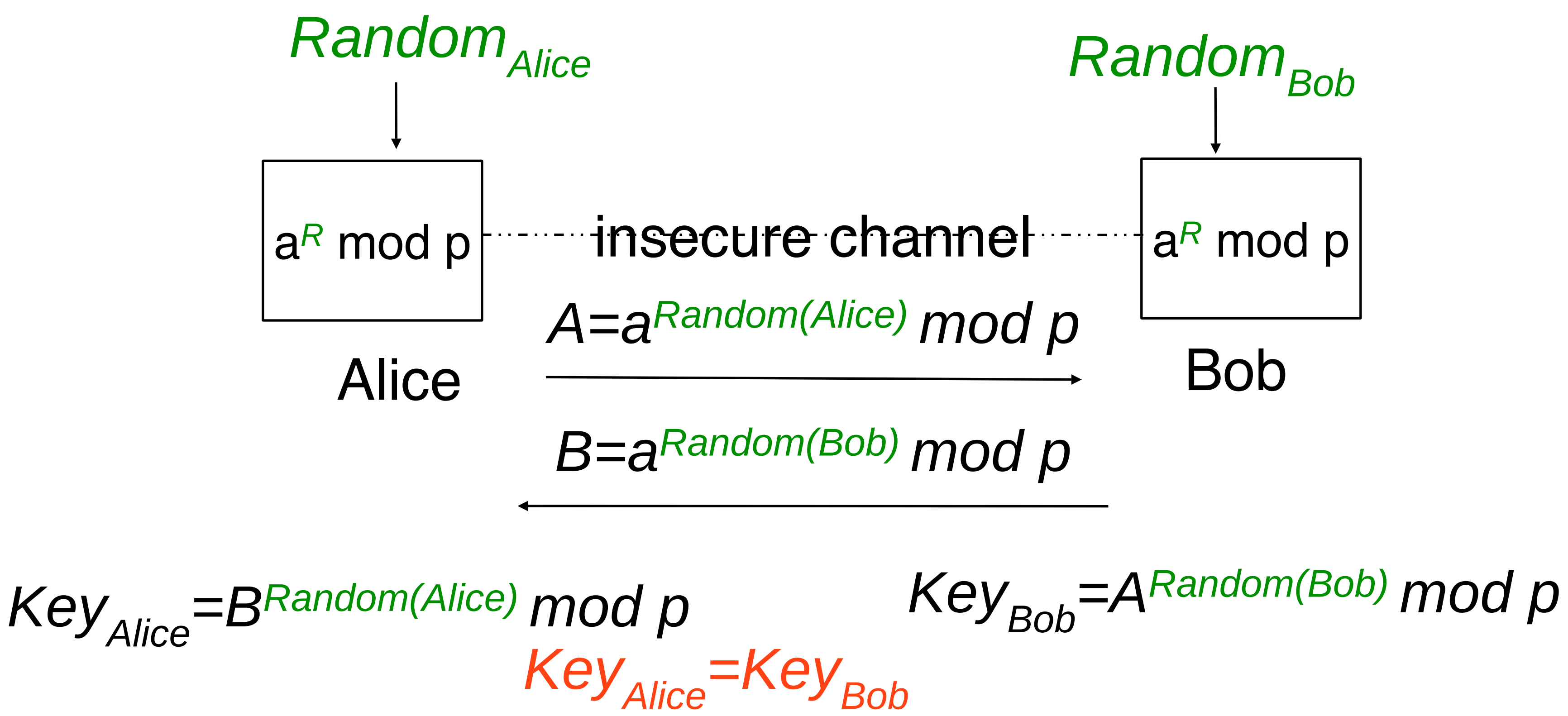
- Does SSL provide PFS ?

Perfect Forward Secrecy (2)

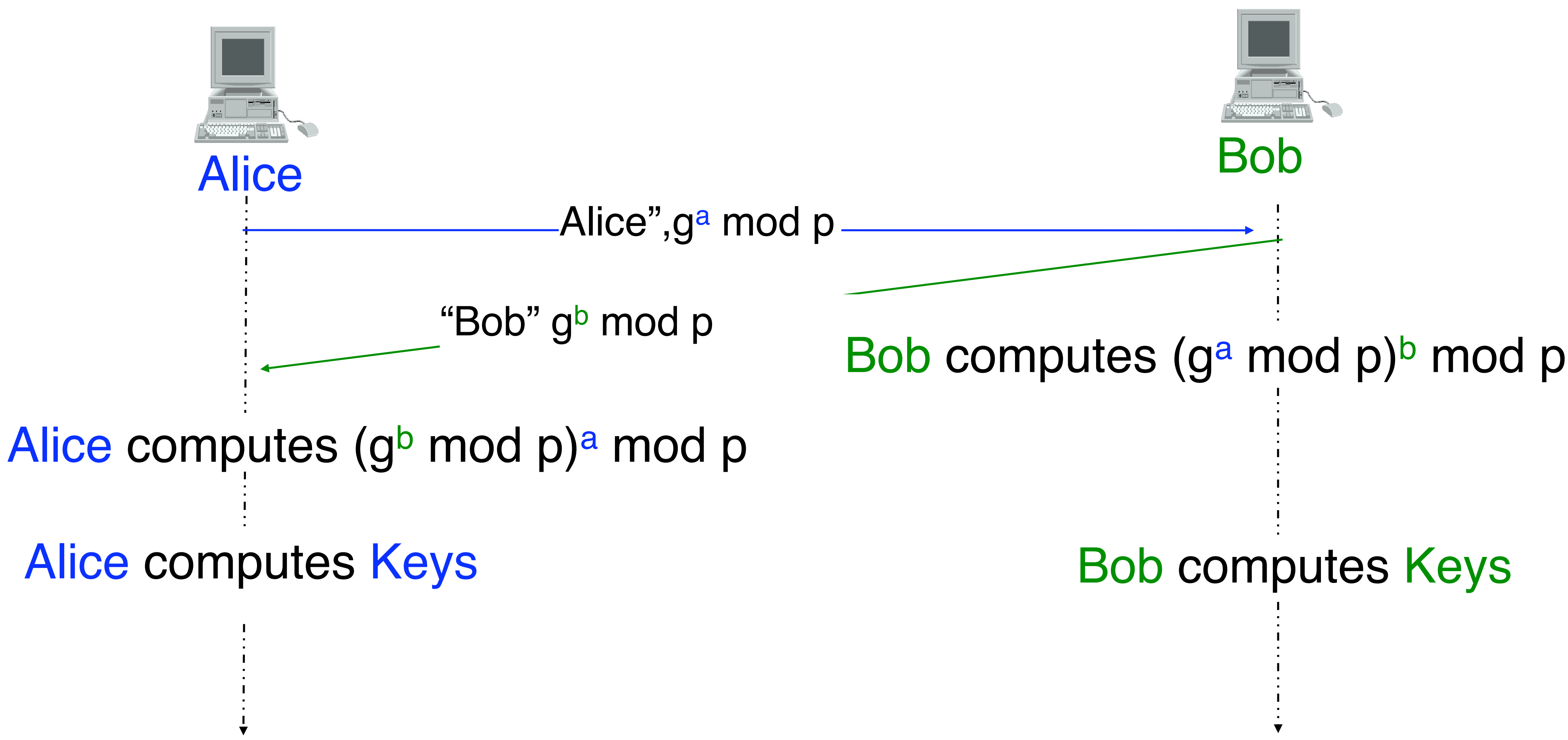
- How to provide PFS ?
  - Compute session keys based on random numbers and never store the session keys after a conversation
    - Session keys should not depend on stored information
  - If the conversation lasts long, regularly change the encryption keys
    - Common good practice for security

Cryptographical building blocks
Diffie Hellman

- Diffie-Hellman key exchange
  - two public numbers known by Alice and Bob
    - a : integer, p : prime



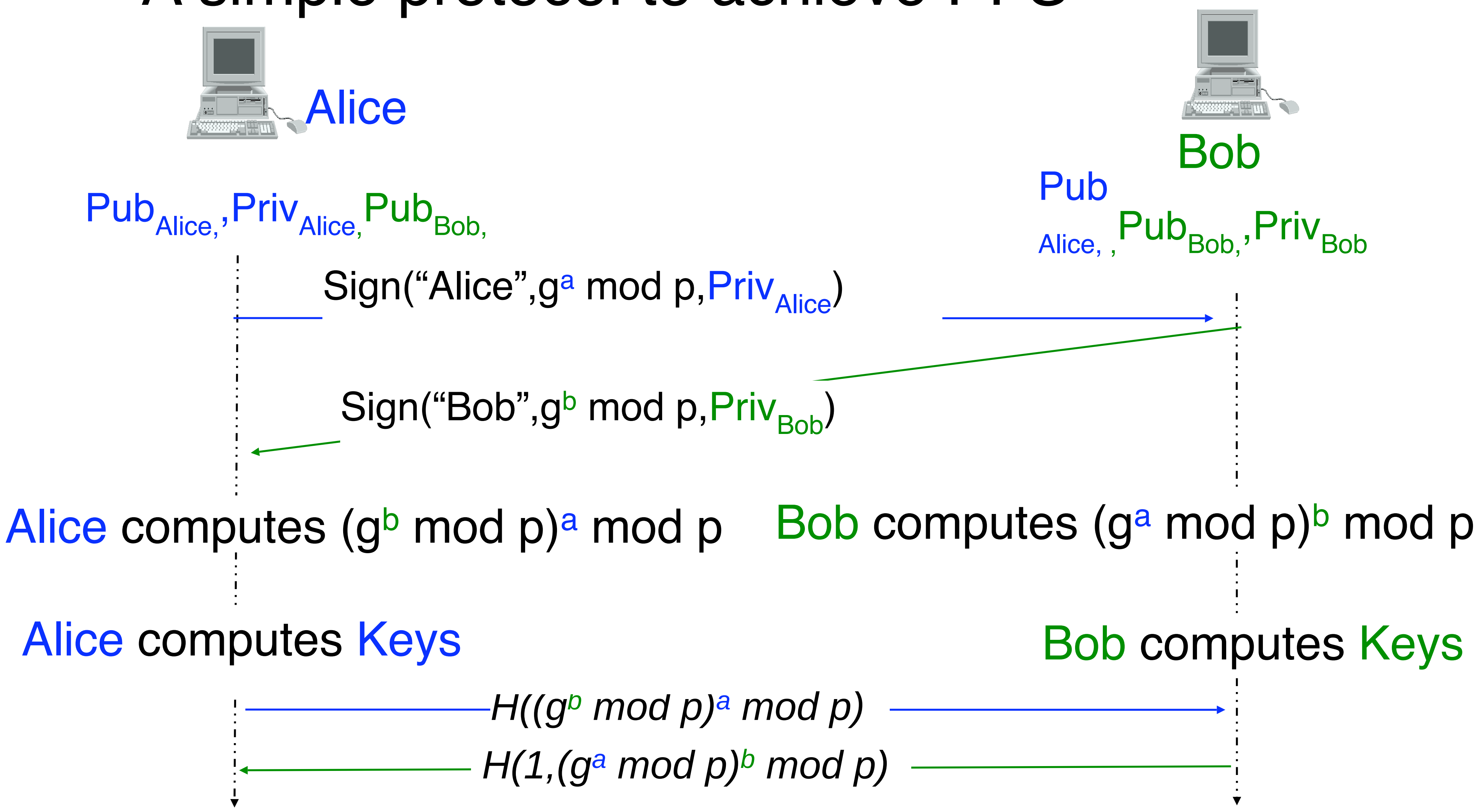
Can we simply reuse Diffie-Hellman ?



- What are the risks ?

How to support Perfect Forward Secrecy

- A simple protocol to achieve PFS



In this example, a and b are random numbers generated by respectively Alice and Bob.

The messages written in *Italics* are encrypted with the session keys derived by Alice and Bob.

Evaluation of simple protocol

- Time to establish a security association
  - Can be used over UDP
  - Fast; one round-trip-time is sufficient

- DoS risk
  - Spoofed packets requesting establishment of a security association could cause a DoS attack on the responder
    - Responder must check signature
    - Responder must perform Diffie-Hellman computation

Evaluation of simple protocol

- Fragmentation risk
  - First message sent by initiator can be large
    - Diffie Hellmann parameters
    - Signature information
  - Message is probably too large for a single IP packet and fragmentation will be required
    - DoS on IP packet reassembly on the responder is possible
      - Hosts have difficulties in correctly supporting reassembling

The fragmentation problem is discussed in :
Charlie Kaufman , Radia Perlman ,Bill Sommerfeld , DoS protection for UDP-based protocols, Proceedings of the 10th ACM conference on Computer and communications security, Washington D.C., USA, 2003

Internet Key Exchange

- Issues to be addressed
  - Transport protocol
    - UDP
      - Lightweight, but retransmissions must be handled by IKE
      - Fragmentation issues to be considered
    - TCP
      - No need to take retransmissions into account in IKE
      - If attacker can break TCP connection, then IPSec won't work
  - Secure channel between initiator and responder
    - In practice, several channels could be required
    - It should be possible to identify the security channels

IKE and ISAKMP are defined in :
RFC2408 Internet Security Association and Key Management Protocol (ISAKMP). D. Maughan, M. Schertler, M. Schneider, J. Turner. November 1998.
RFC2409 The Internet Key Exchange (IKE). D. Harkins, D. Carrel. November 1998.

IKE has been simplified and improved. The new version is more readable and is described in :

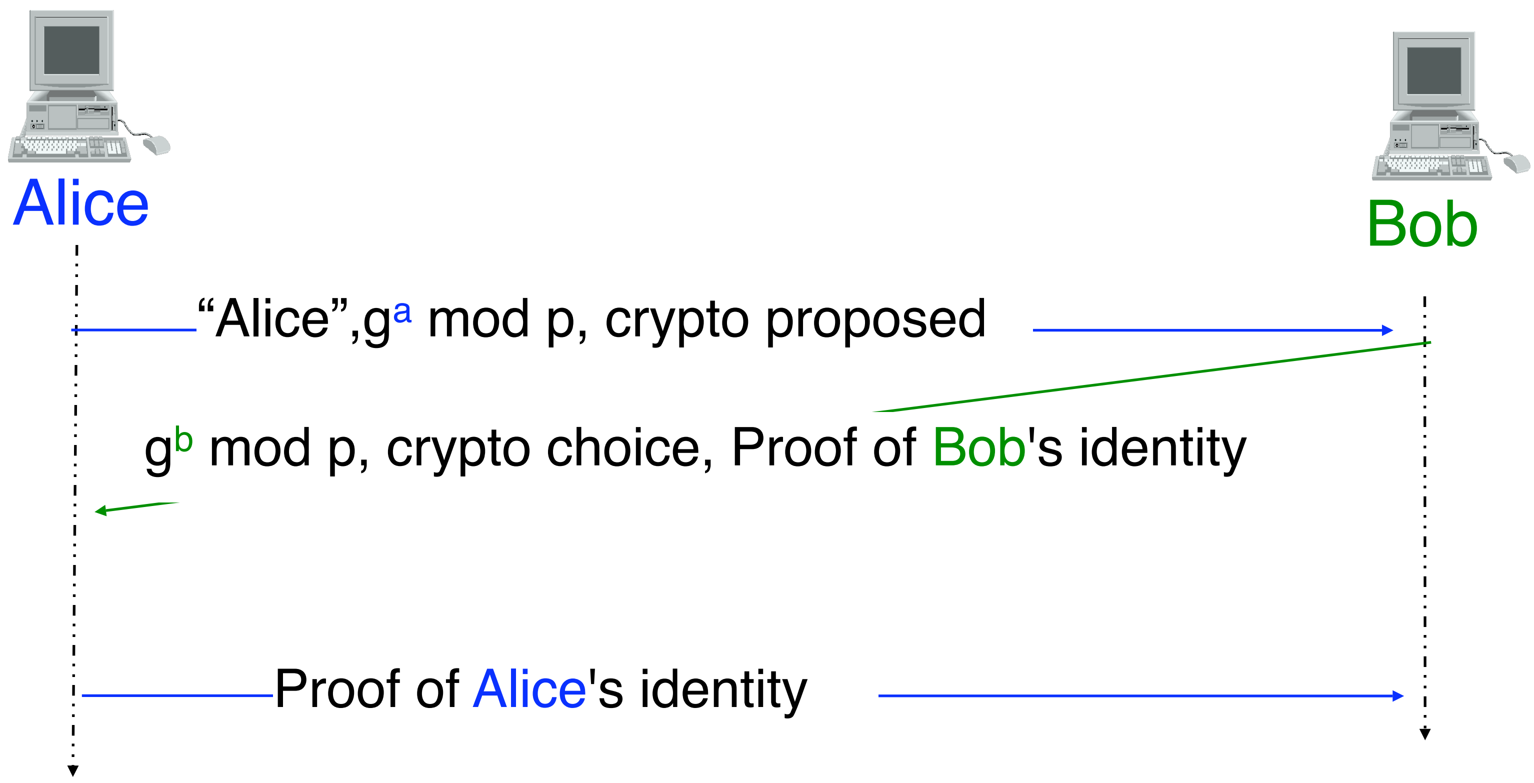
C. Kaufman, Ed., Internet Key Exchange (IKEv2) Protocol, RFC4306, December 2005

Internet Key Exchange

- Elements of the protocol
  - Negotiation of the cryptographic algorithms to be used over the secure session
    - There are many possible encryption and authentication algorithms that could be used
  - Computation of session keys
    - Initiator and responder must compute the same key
  - Proof of identities
    - Responder wants to verify identity of initiator and vice-versa

Internet Key Exchange v1 phase 1

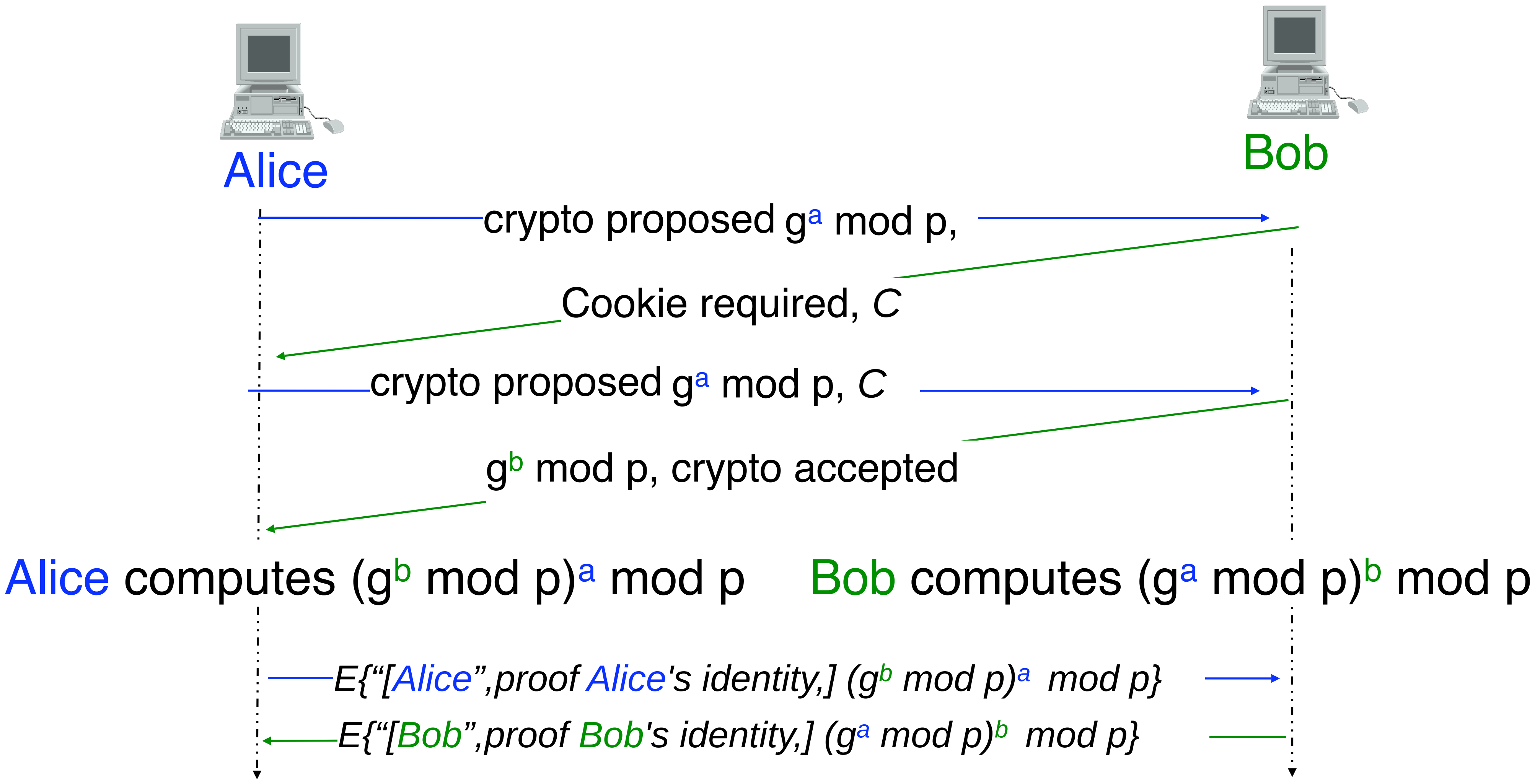
- First solution
  - Based on Diffie-Hellmann
  - three messages exchanged, risk of DoS



This corresponds to the aggressive mode of IKEv1. In this mode, Alice assumes the crypto that Bob will supports and selects a Diffie-Hellman group (g,p). If Bob does not support this group, then the establishment of the security association will fail.

Internet Key Exchange v2

- Principle of the protocol



This is a simplified version of IKEv2. In reality, Alice and Bob derive encryption and authentication keys from the DH key. These keys are used to encrypt and authenticate the messages. Additional details about IKEv2 may be found in RFC4306.

A tutorial on IKEv2 may be found in
R. Perlman, Understanding IKEv2 : Tutorial, and rationale for decisions, draft-ietf-ipsec-ikev2-tutorial-01.txt, internet draft, work in progress, Feb. 200

The crypto proposed follows the same approach as with SSL by using suites of crypto mechanisms in IKEv2 (authentication, encryption, ...)
The cookie is usually computed based on a hash to allow Bob to remain stateless until the reception of the third message.
The keys derived by Alice and Bob are different for each direction and for encryption and authentication. Furthermore, the keys depend on the a and b values but also on the cookie chosen by Bob. This allows Bob to use several times the same diffie hellman values without breaking PFS since the keys in different sa will be different.

Services provided by IPSec

- Authentication and data integrity only
  - AH : Authentication Header
  - Principle
    - Sender authenticates source and protects packet content
    - Packet content can be read by anyone
    - Destination authenticates packet source and content

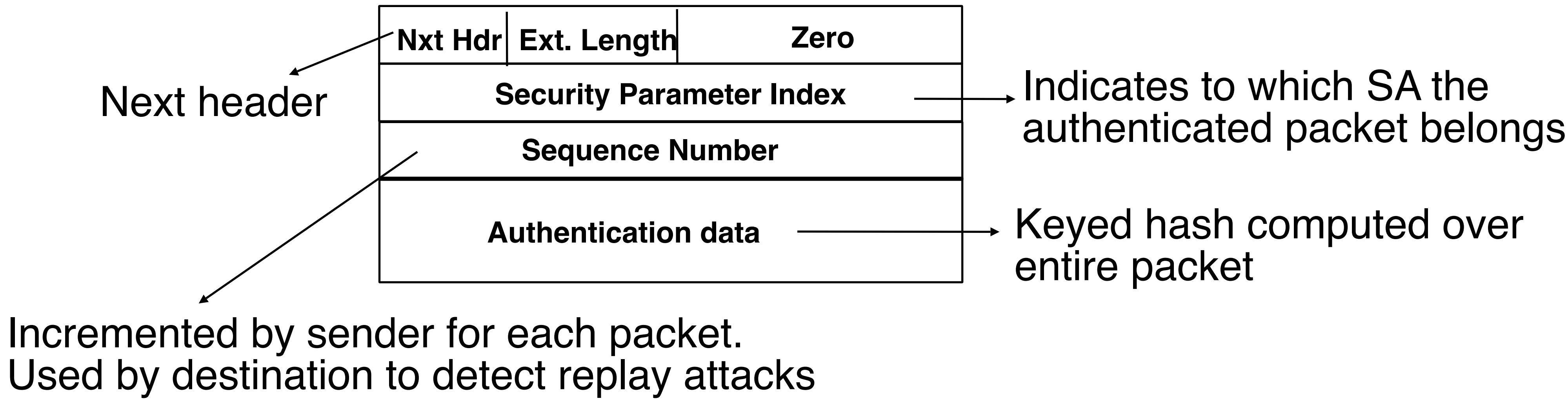
The AH header is defined in
RFC2402 IP Authentication Header. S. Kent, R. Atkinson. November 1998.

The new version is described in :

RFC4302, IP Authentication Header. S. Kent, December 2005

The AH header

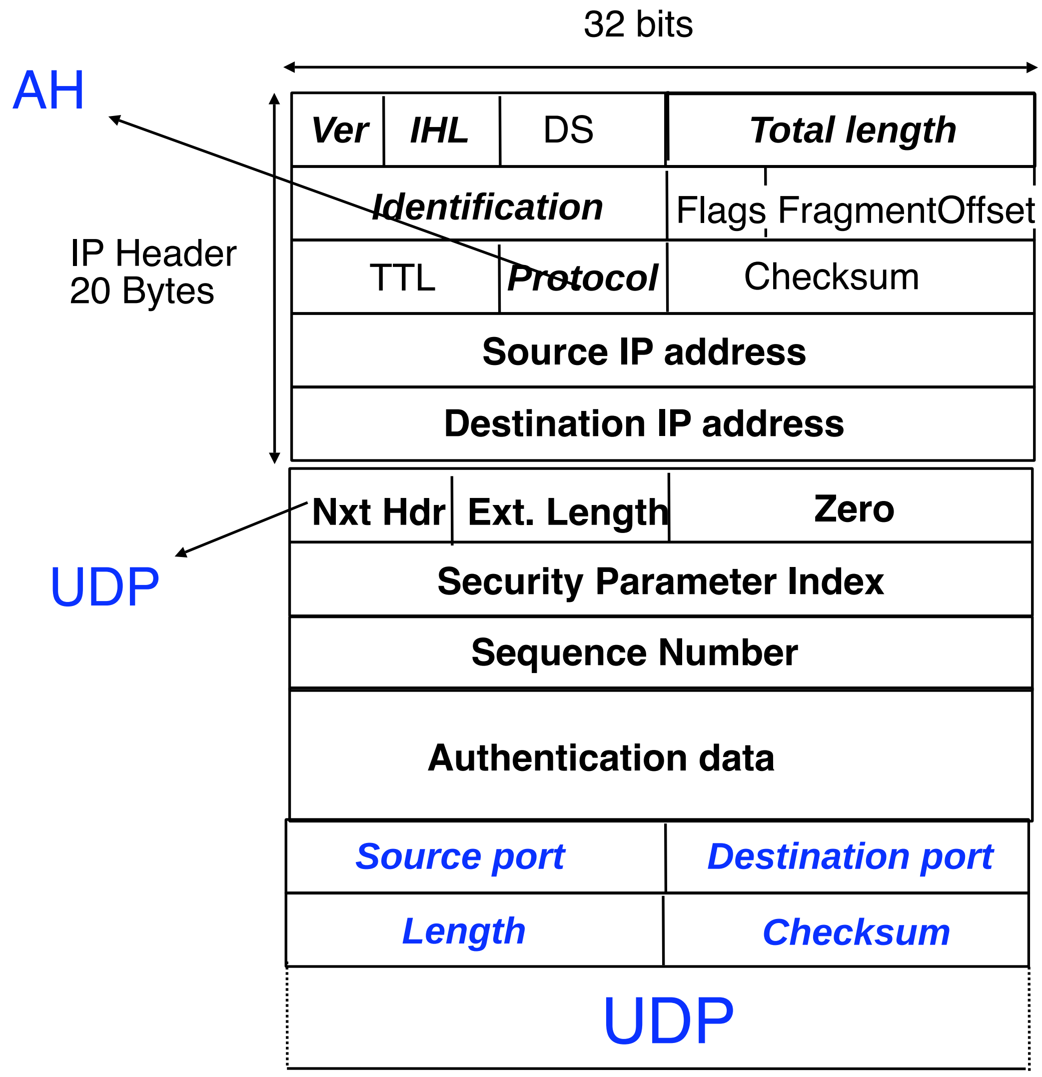
- Formatted as an IPv6 option



- How to compute the authentication data ?
  - Keyed Hash computed over payload and immutable fields in packet header

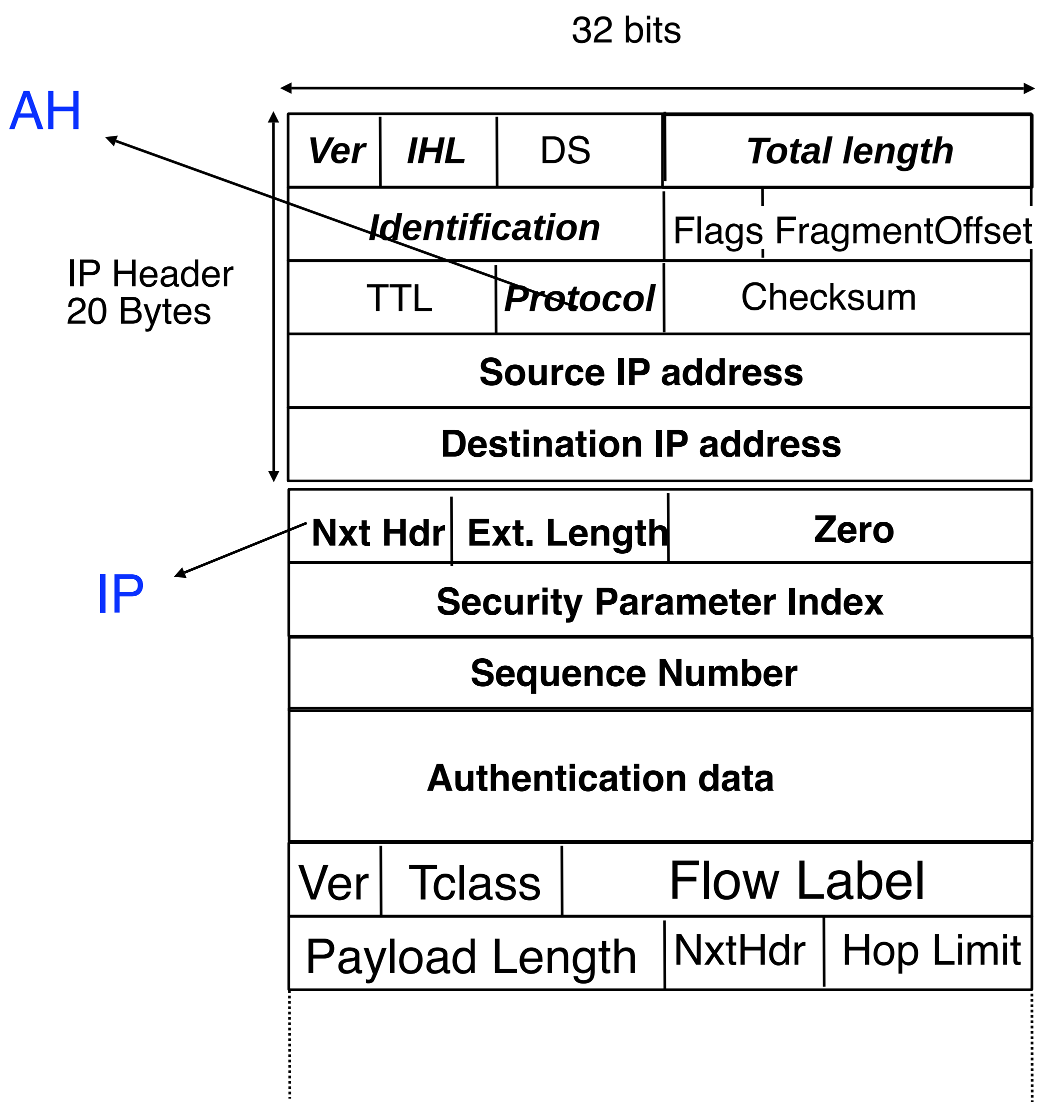
The immutable fields are the fields of the IP header that are not changed by intermediate routers. These include Version, Total Length, Header Length, Identification, Protocol, Source address, Destination address and packet payload
The DSCP, TTL, Fragment Offset, flags and checksum are not used to compute authentication because their value may change inside network

AH Transport mode



The fields that appear in bold italics are those that are used by the source to compute the authentication header. When source routing is used, the utilization of the destination address in the computation of the authentication data is more complex, but source routing should rarely be used in practice.

AH Tunnel mode

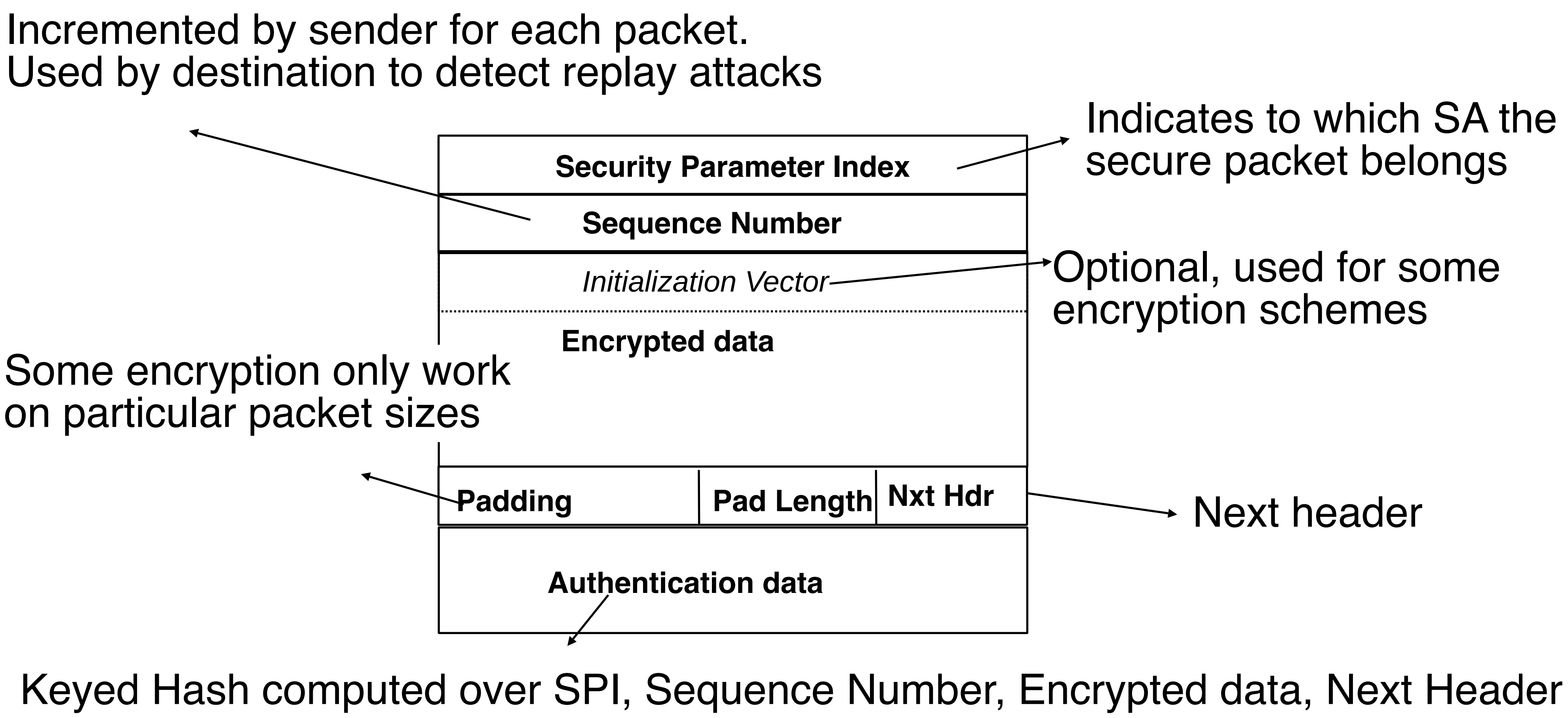


Services provided by IPSec

- Encryption and data integrity
  - ESP : Encapsulating Security Payload
  - Principle
    - Sender authenticates and encrypts IP packet
      - entire packet in tunnel mode
      - packet payload (including transport headers) in transport mode
    - Packet content cannot be read by intermediate nodes
    - Destination decrypts and checks authenticity of packet

The ESP Protocol

- Principle
  - Encrypts and authenticates
    - payload in transport mode, entire packet in tunnel mode



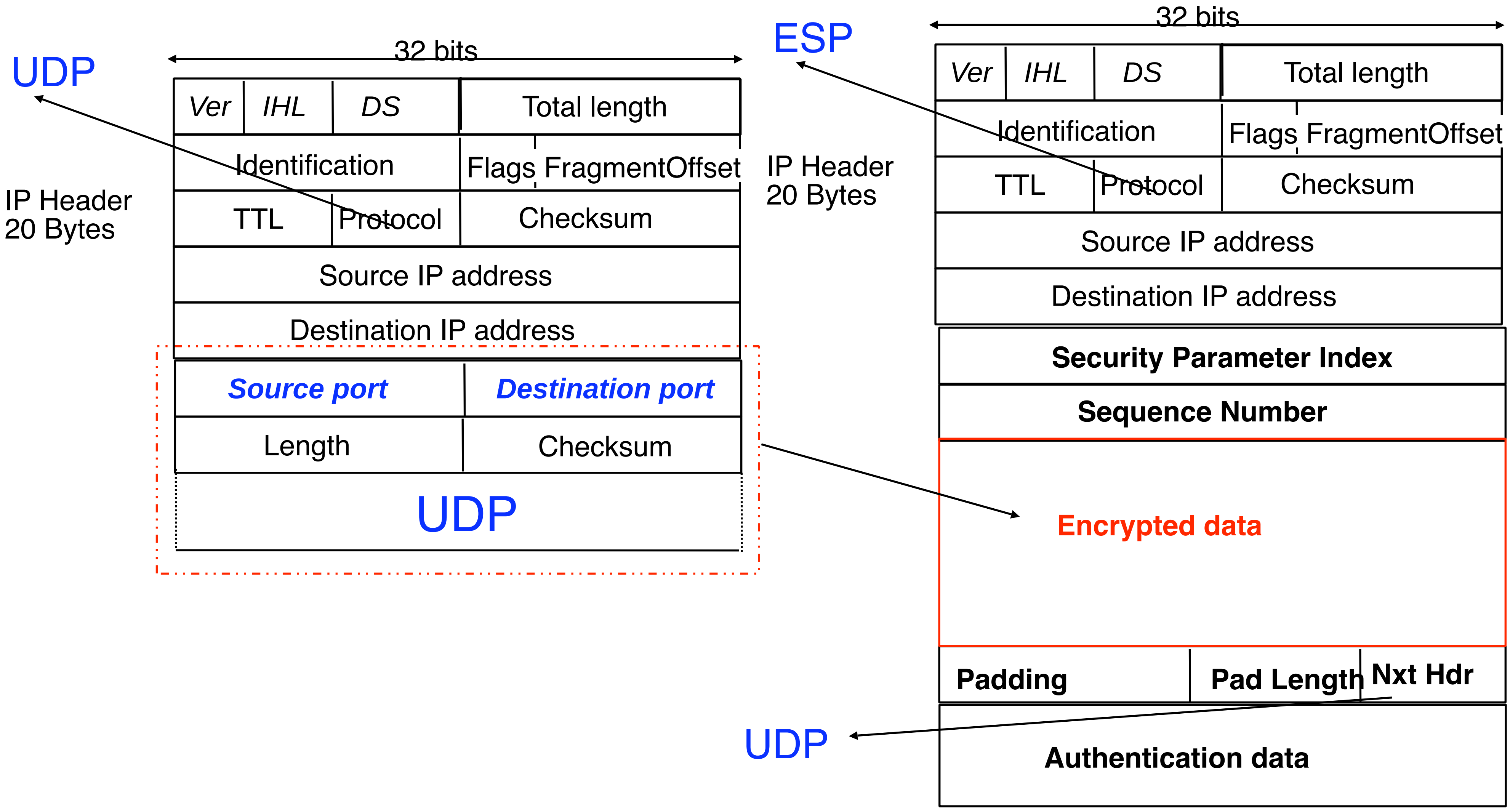
The ESP protocol is defined in :

RFC2406 IP Encapsulating Security Payload (ESP). S. Kent, R. Atkinson. November 1998.

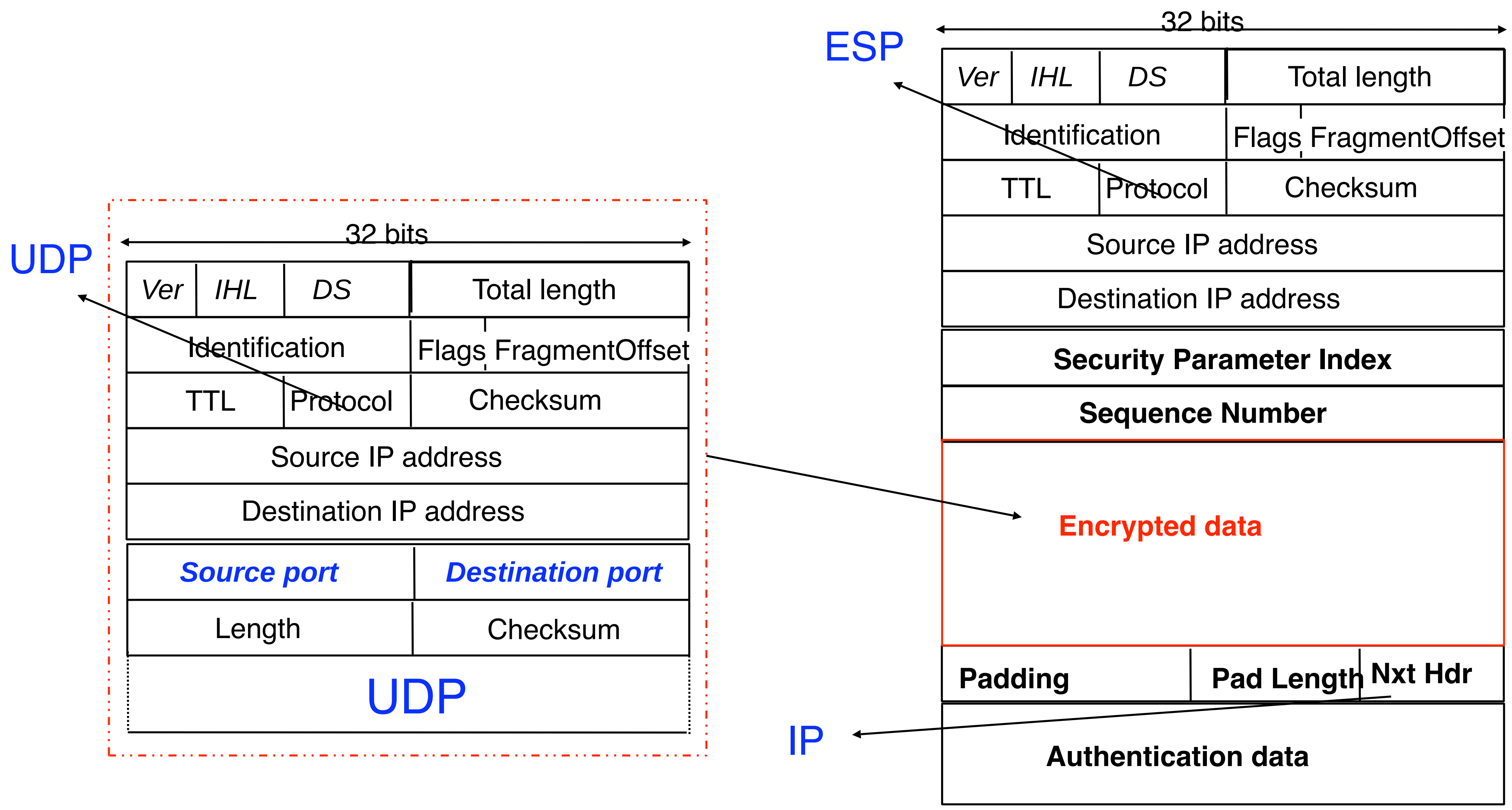
This document has been updated in :

RFC4303, IP Encapsulating Security Payload (ESP). S. Kent, December 2005

ESP : Transport Mode



ESP : Tunnel Mode



AH versus ESP

AH

- Provides authentication
- No secrecy
- Hardware implementation
  - Authentication data must be placed inside header after computation
- Firewall traversal
  - Firewall sees transport-level information
- Paranoiac government
  - packets are not encrypted and eavesdropping is still possible

ESP

- Provides authentication
- Provides secrecy
- Hardware implementation
  - On-the-fly encryption and authentication are possible since authentication data is placed inside trailer
- Firewall traversal
  - Firewall does not see transport-level headers, difficult to use null-encryption to solve this problem
- Paranoiac government
  - ESP packets can be detected and blocked ?

Do we really need both AH and ESP ?

Internet and Network security

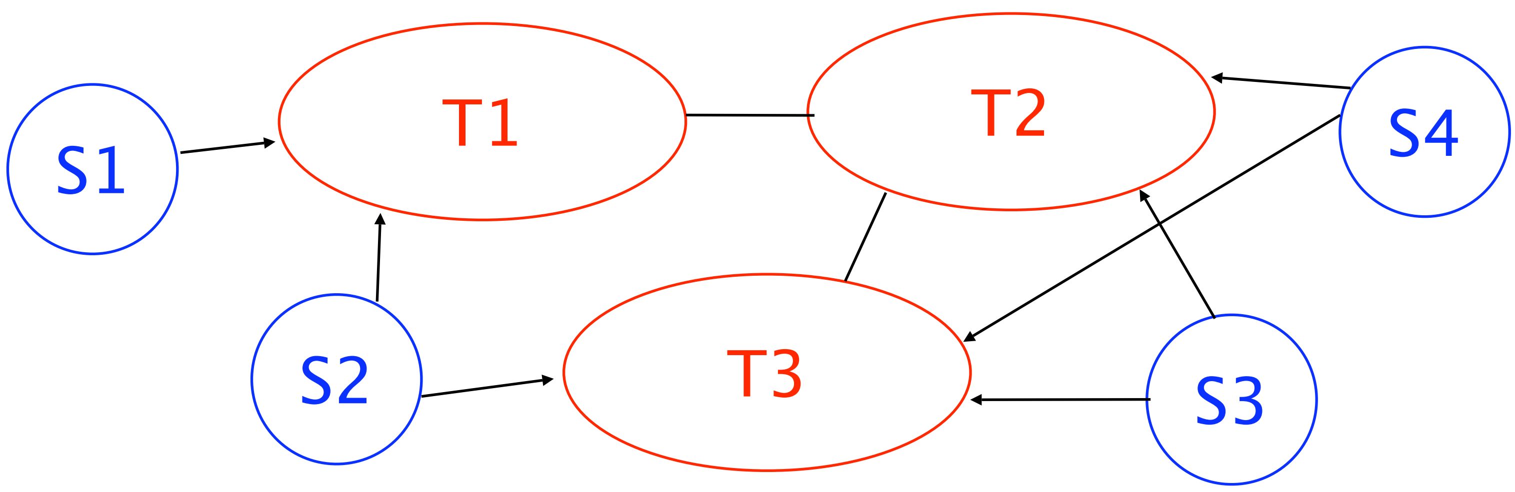
- Crypto building blocks
- Application-layer security
- Transport-layer security
- Network-layer security
  - IPv4
  - IPv6
  - IPSec
  - Routing security

Organisation of the Internet

- Internet is composed of more than 28.000 autonomous routing domains
  - A domain is a set of routers, links, hosts and local area networks under the same administrative control
    - A domain can be very large...
      - AS568: SUMNET-AS DISO-UNRRA contains 73154560 IP addresses
    - A domain can be very small...
      - AS2111: IST-ATRIUM TE Experiment a single PC running Linux...
  - Domains are interconnected in various ways
    - The interconnection of all domains should in theory allow packets to be sent anywhere
    - Usually a packet will need to cross a few ASes to reach its destination

Types of domains

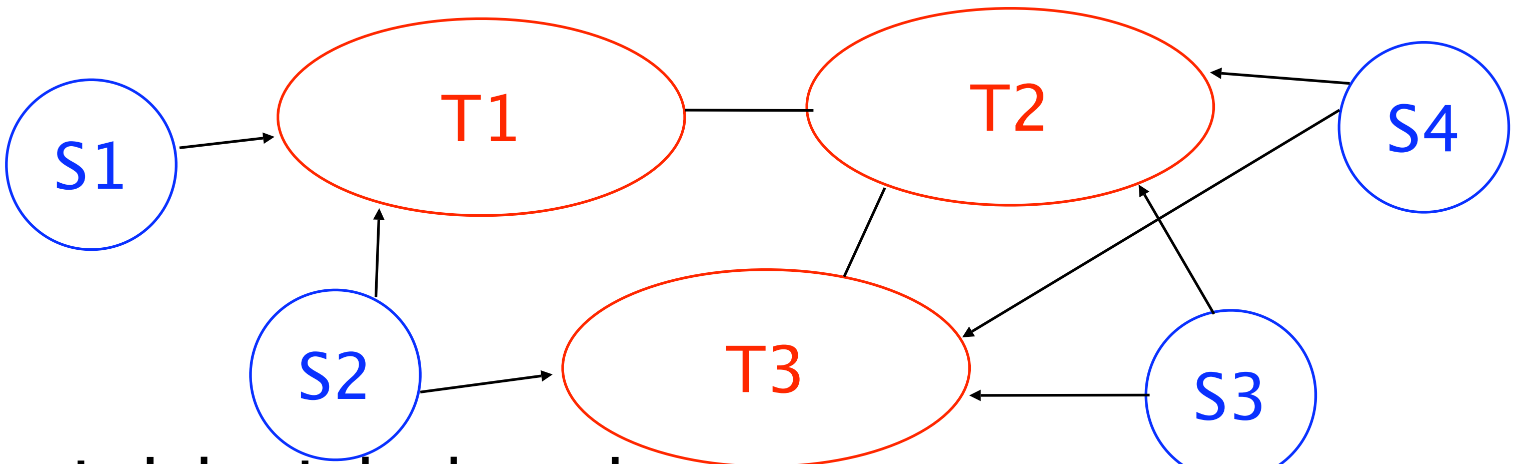
- Transit domain
  - A transit domain allows external domains to use its own infrastructure to send packets to other domains



- Examples
  - UUNet, OpenTransit, GEANT, Internet2, RENATER, EQUANT, BT, Telia, Level3,...

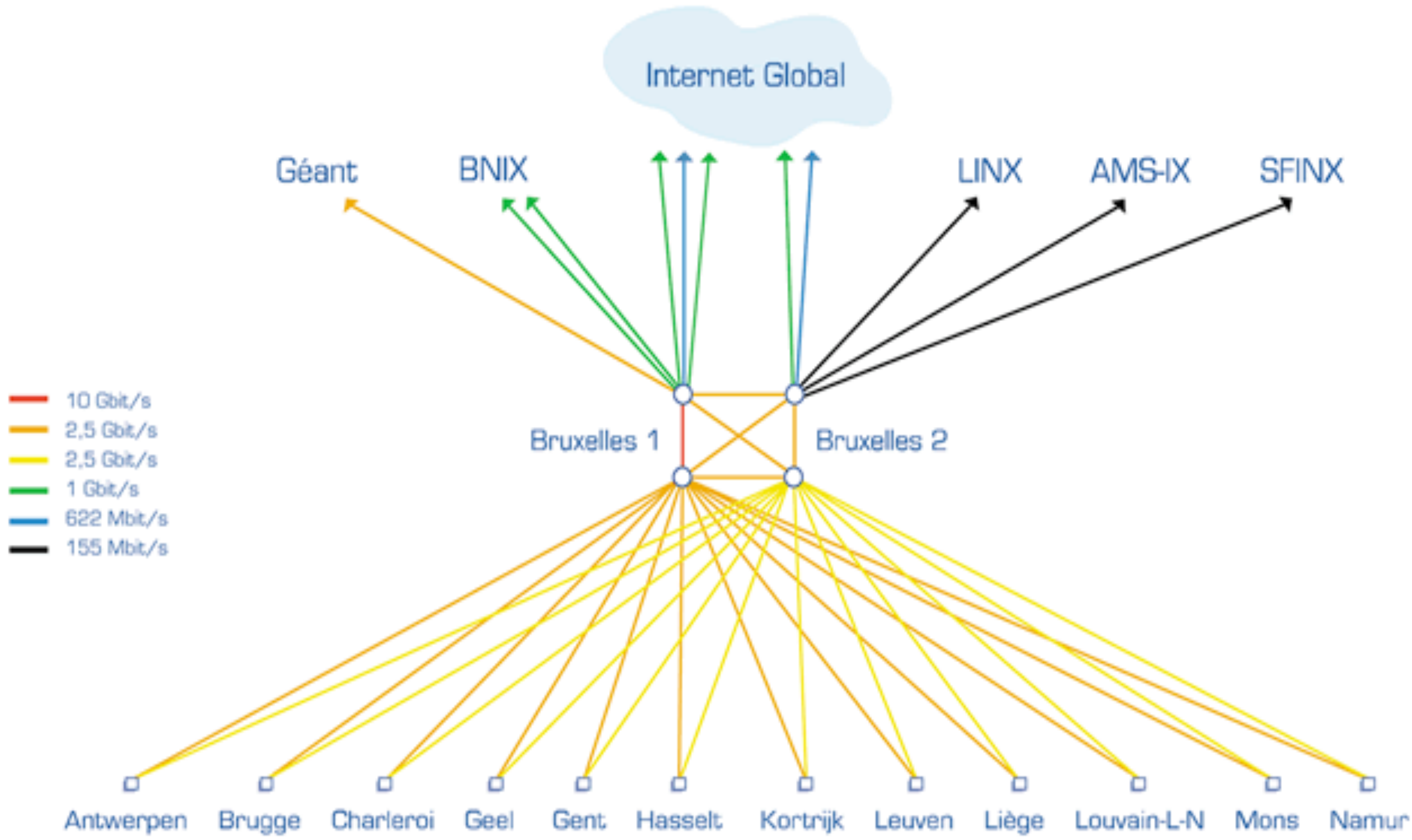
Types of domains (2)

- Stub domain
  - A stub domain does not allow external domains to use its infrastructure to send packets to other domains
    - A stub is connected to at least one transit domain
      - Single-homed stub : connected to one transit domain
      - Dual-homed stub : connected to two transit domains



  - Content-rich stub domain
    - Large web servers : Yahoo, Google, MSN, TF1, BBC,...
  - Access-rich stub domain
    - ISPs providing Internet access via CATV, ADSL, ...

A Stub domain : Belnet

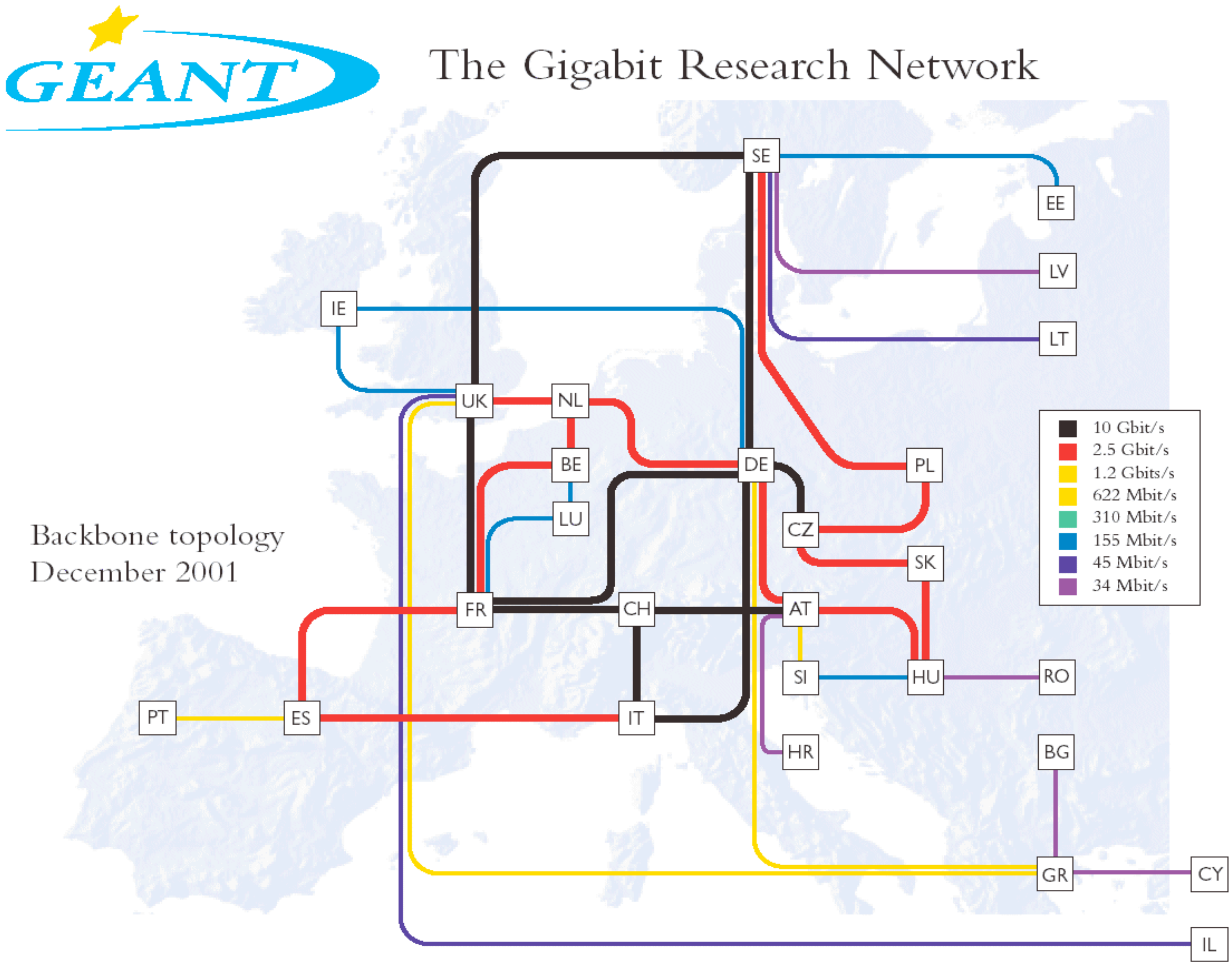


BGP/2003.1.

Source : http://www.belnet.be

Other maps of ISPs may be found at :
http://www.cs.washington.edu/research/networking/rocketfuel/interactive/

A transit domain : GEANT



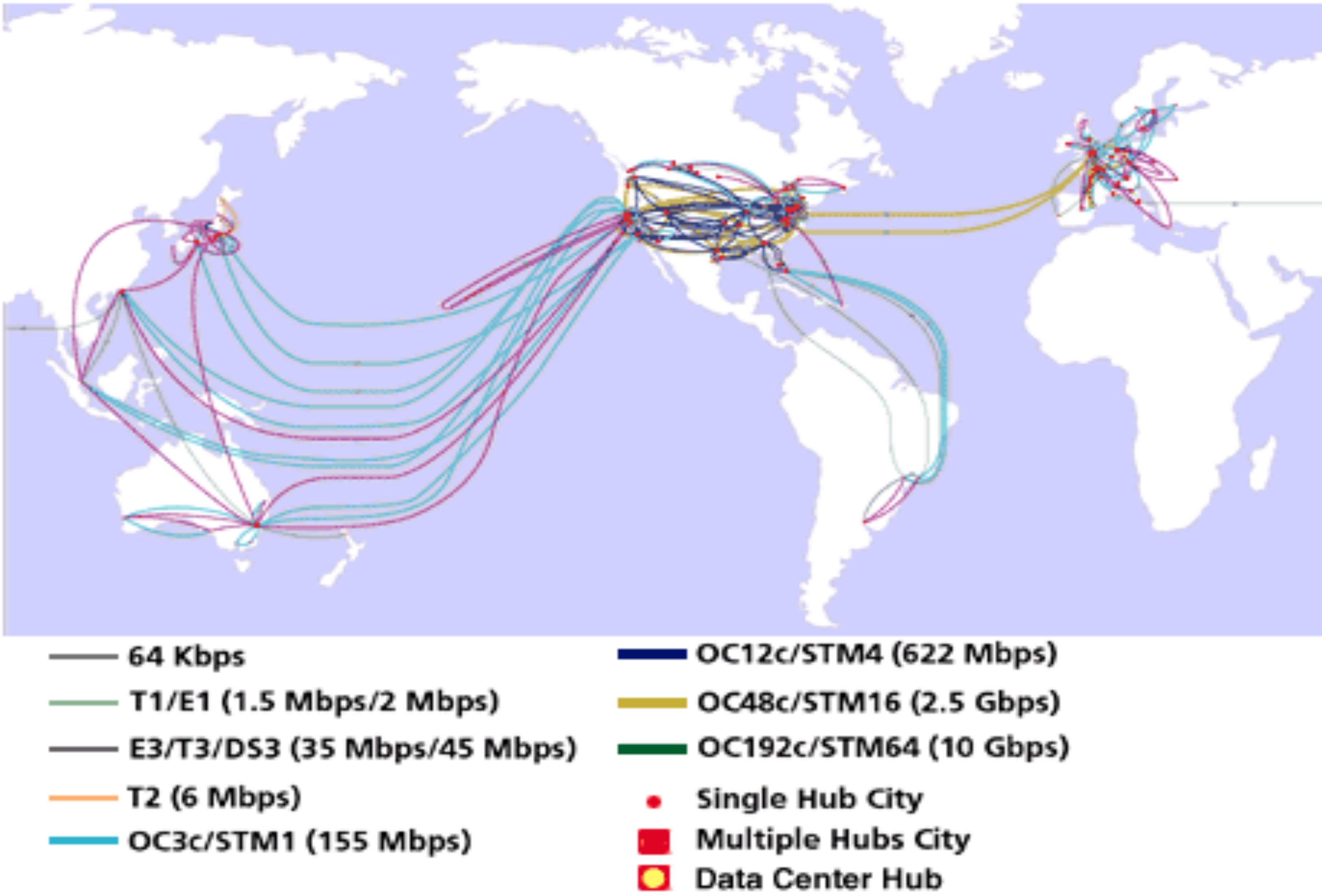
BGP/2003.1.

Source http://www.dante.net

A transit domain : BT/IGnite

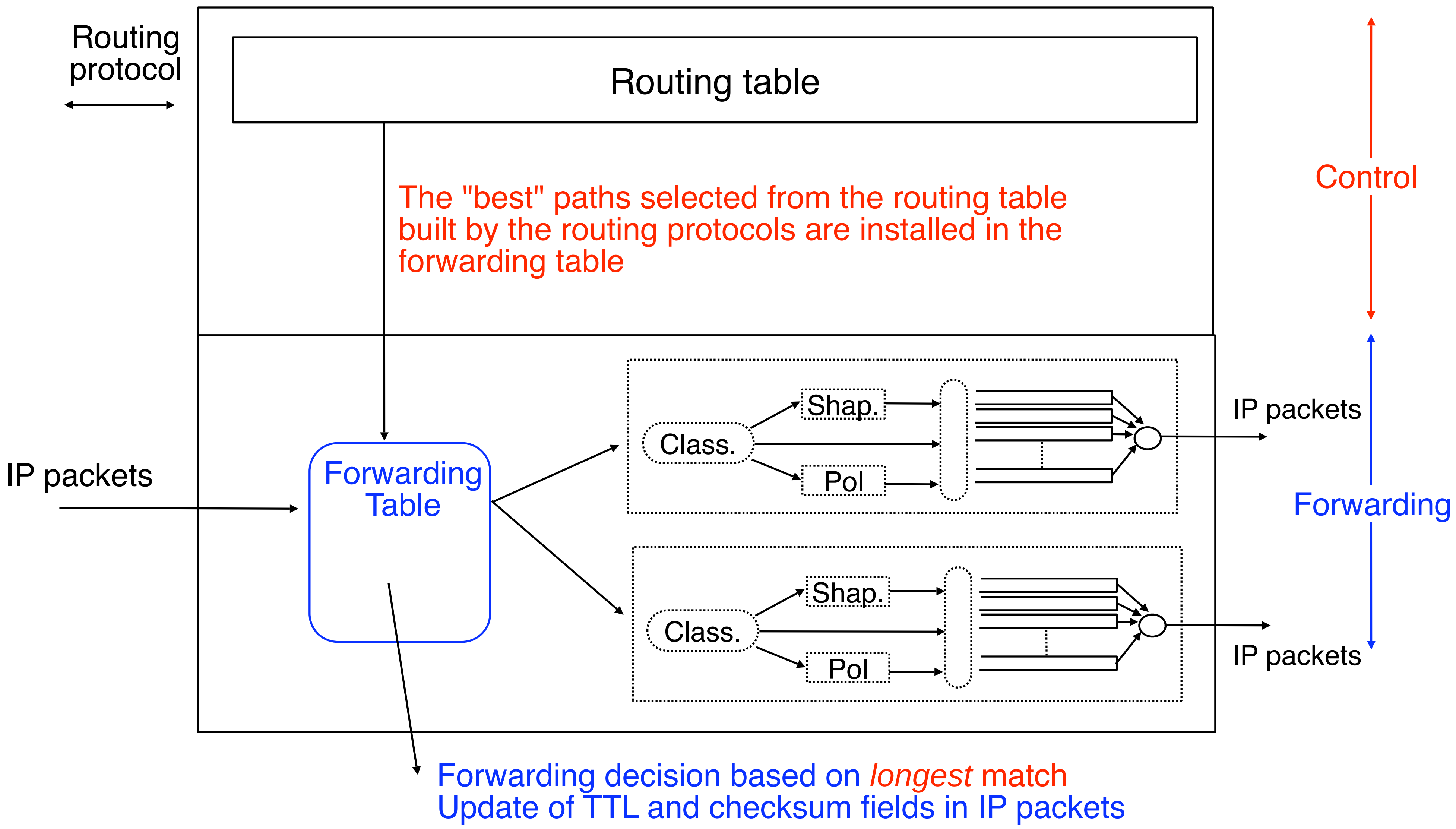
Source : http://www.ignite.net/info/maps.shtml

A large transit domain : UUNet



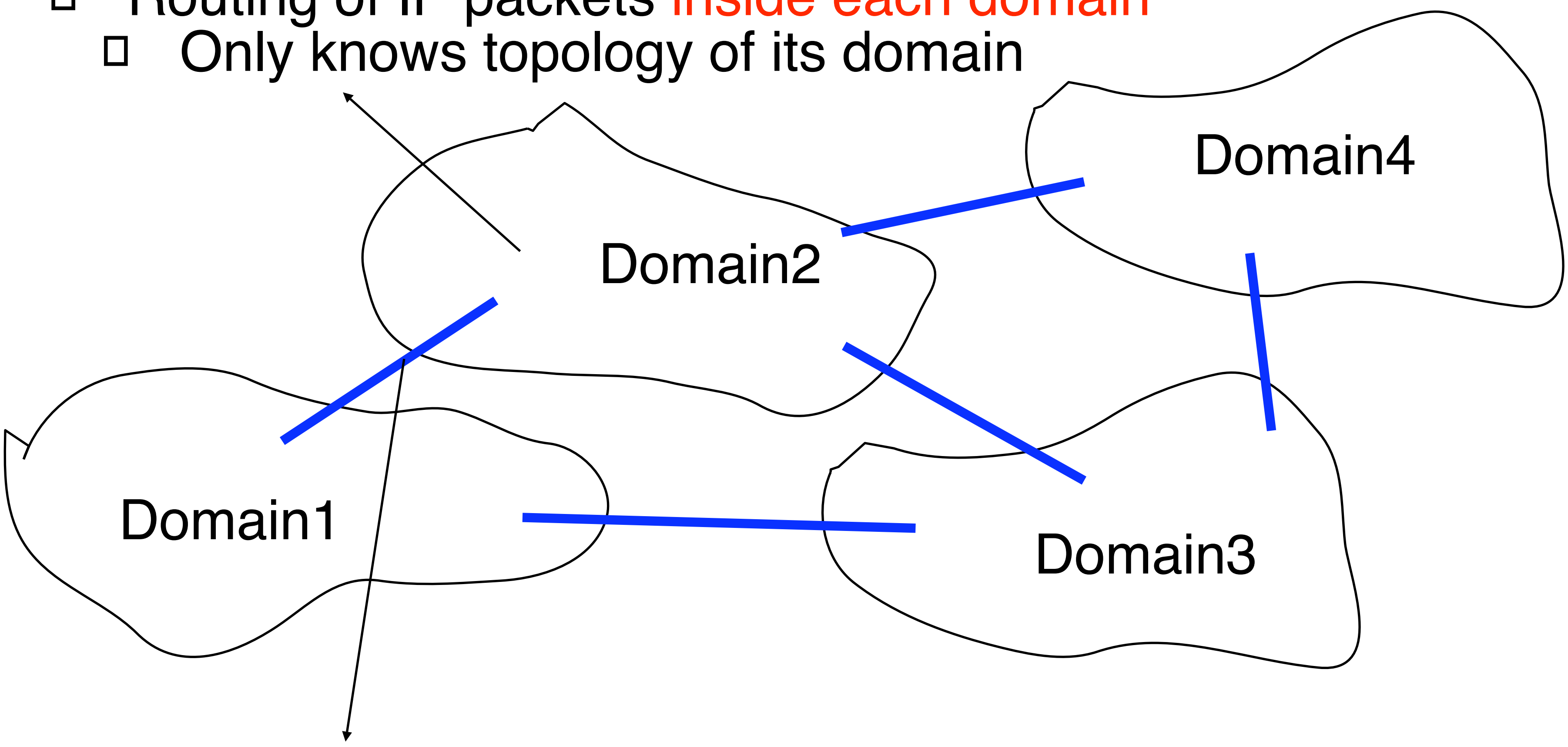
Source http://www.uu.net

Architecture of a normal IP router



Internet routing

- Interior Gateway Protocol (IGP)
  - Routing of IP packets inside each domain
    - Only knows topology of its domain



- Exterior Gateway Protocol (EGP)
  - Routing of IP packets between domains
    - Each domain is considered as a blackbox

Intradomain routing

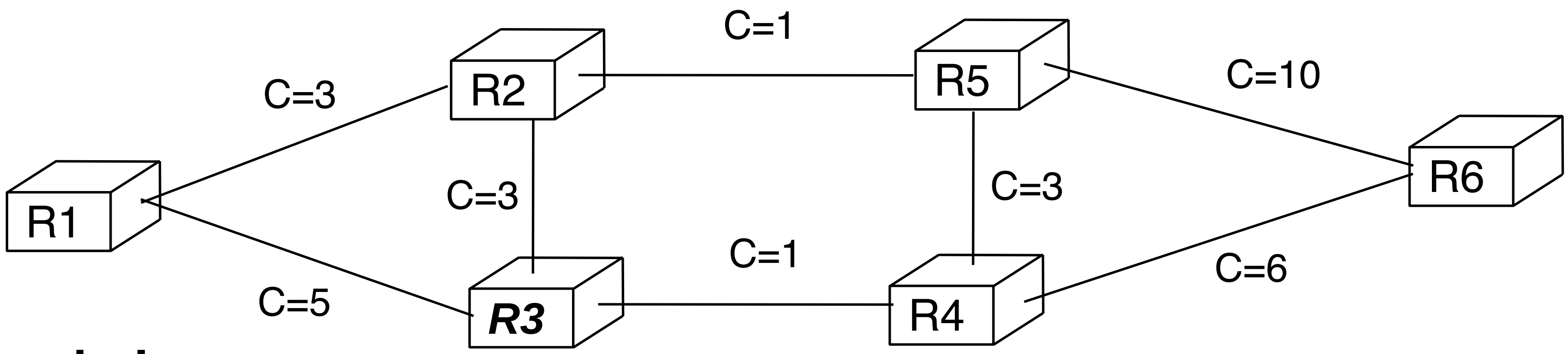
- Goal
  - Allow routers to transmit IP packets along the best path towards their destination
    - best usually means the shortest path
      - Shortest measured in seconds or as number of hops
    - sometimes best means the less loaded path
  - Allow to find alternate routes in case of failures

- Behaviour
  - All routers exchange routing information
    - Each domain router can obtain routing information for the whole domain
    - The network operator or the routing protocol selects the cost of each link

Three types of Interior Gateway Protocols

- Static routing
  - Only useful in very small domains
- Distance vector routing
  - Routing Information Protocol (RIP)
    - Still widely used in small domains despite its limitations
- Link-state routing
  - Open Shortest Path First (OSPF)
    - Widely used in enterprise networks
  - Intermediate System- Intermediate-System (IS-IS)
    - Widely used by ISPs

Distance vector routing



- Principle
  - Router configuration
    - Cost associated with each link
  - Each router sends <u>periodically</u> a distance vector containing, for each known prefix, :
    1. The IP prefix
    2. The distance between itself and the destination
       - The distance vector is a summary of the router's routing table
  - Each router receives its neighbour's distance vectors and builds its routing table based on those vectors

Issues with distance vector routing

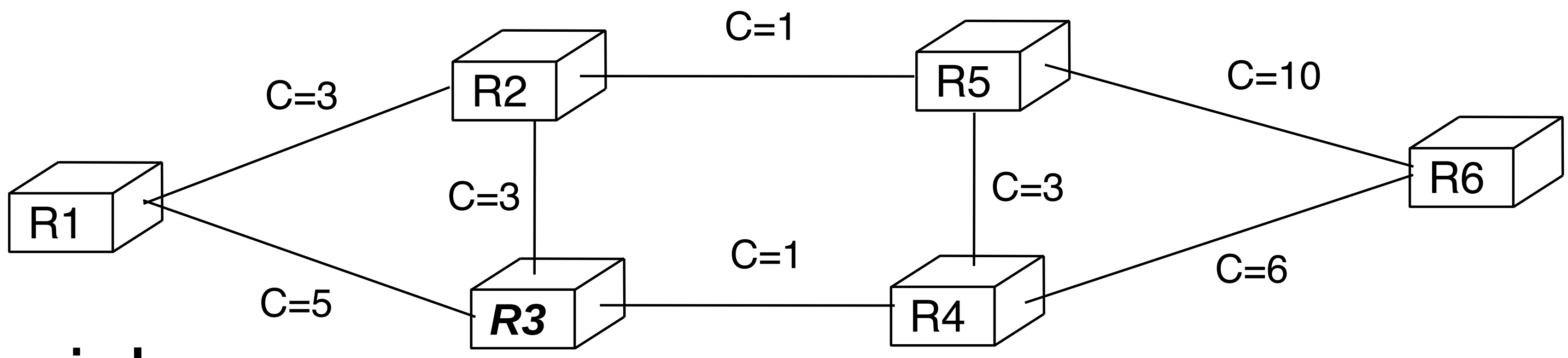
- How to deal with link failures ?
  - Routers should send their distance vector when they detect the failure of one of their links

- How to avoid the count-to-infinity problem ?
  - Utilise a non-redundant star shaped network

  - Limit the maximum distance between routers
    - For RIP, $\infty = 16$!

  - Split horizon
    - Router A does not advertise to router B the routes for which it sends packets via router B
  - Split horizon with Poison reverse

Link state routing



- Principle
  - Each router builds link state packet containing its local topology
    - Link state packets are created at regular intervals and when the local topology changes
  - Link state packet is reliably flooded to all routers inside the domain
  - Each router knows the complete domain topology
  - Computes routing tables by using Dijkstra
    - The best path is the path with the smallest cost

For a description of OSPF, see J. Moy, OSPF : anatomy of an Internet routing protocol, Addison-Wesley, 1998

ISIS is defined in

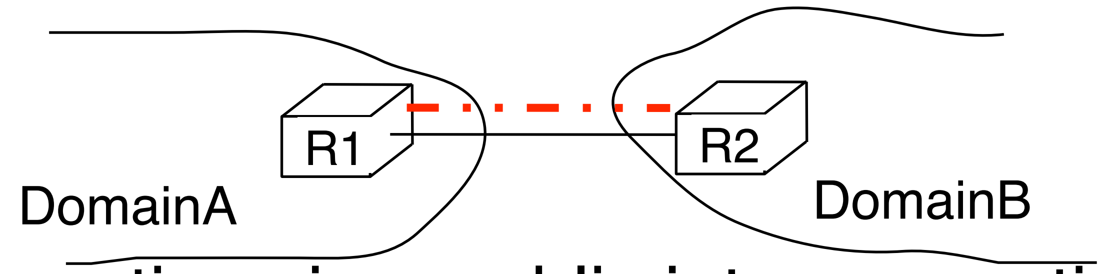
R. Callon, Use of OSI IS-IS for Routing in TCP/IP and Dual Environments, RFC1195, Dec. 1990

Interdomain routing

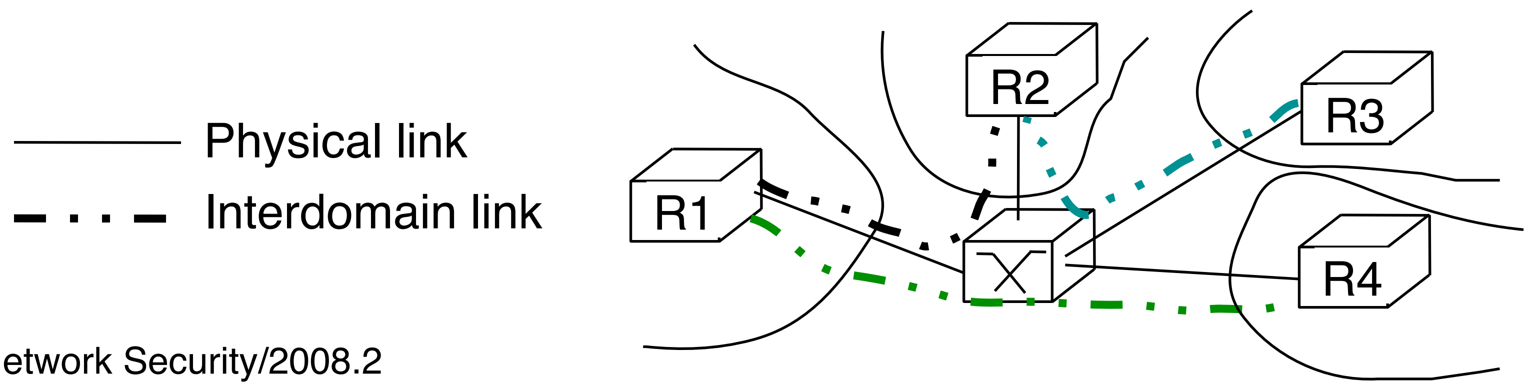
- Goals
  - Allow to transmit IP packets along the best path towards their destination through several transit domains while taking into account the routing policies of each domain without knowing the detailed topology of those domains
    - From an interdomain viewpoint, best path often means *cheapest* path
    - Each domain is free to specify inside its routing policy the domains for which it agrees to provide a transit service and the method it uses to select the best path to reach each destination

Types of interdomain links

- Two types of interdomain links
  - Private link
    - Usually a leased line between two routers belonging to the two connected domains



  - Connection via a public interconnection point
    - Usually Gigabit or higher Ethernet switch that interconnects routers belonging to different domains



For more information on the organization of the Internet, see :

G. Huston, Peerings and settlements, Internet Protocol Journal, Vol. 2, N1 et 2, 1999,
http://www.cisco.com/warp/public/759/ipj_Volume2.html

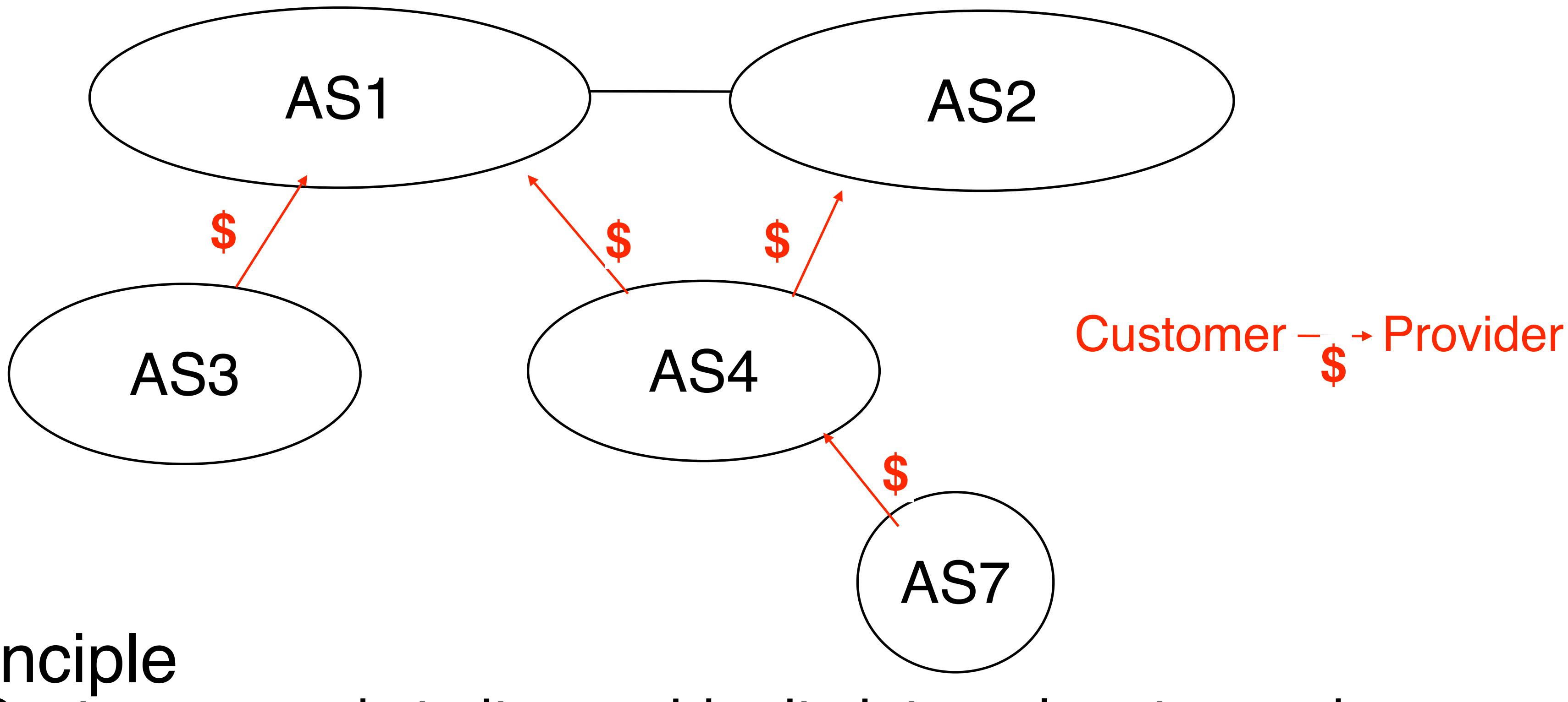
For more information on interconnection points or Internet exchanges, see :

http://www.euro-ix.net/
http://www.ripe.net/ripe/wg/eix/index.html
http://www.ep.net/ep-main.html

Routing policies

- In theory BGP allows each domain to define its own routing policy...
- In practice there are two common policies
  - customer-provider peering
    - **Customer c** buys Internet connectivity from **provider P**
  - shared-cost peering
    - **Domains x** and **y** agree to exchange packets by using a direct link or through an interconnection point

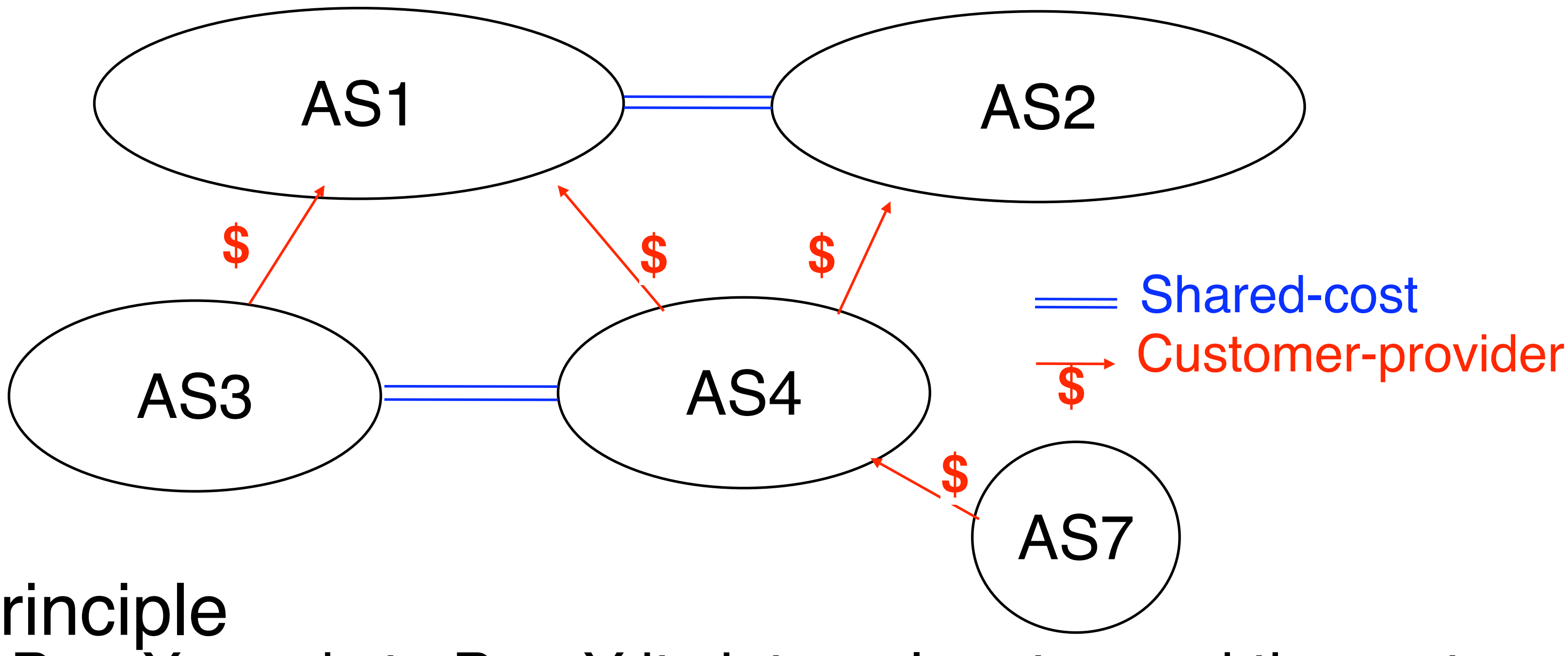
Customer-provider peering



- Principle
  - Customer sends to its provider its internal routes and the routes learned from its own customers
    - Provider will advertise those routes to the entire Internet to allow anyone to reach the Customer
  - Provider sends to its customers all known routes
    - Customer will be able to reach anyone on the Internet

On link AS7-AS4
AS7 advertises its own routes to AS4
AS4 advertises to AS7 the routes that allow to reach the entire Internet
On link AS4-AS2
AS4 advertises its own routes and the routes belonging to AS7
AS2 advertises the routes that allow to reach the entire Internet

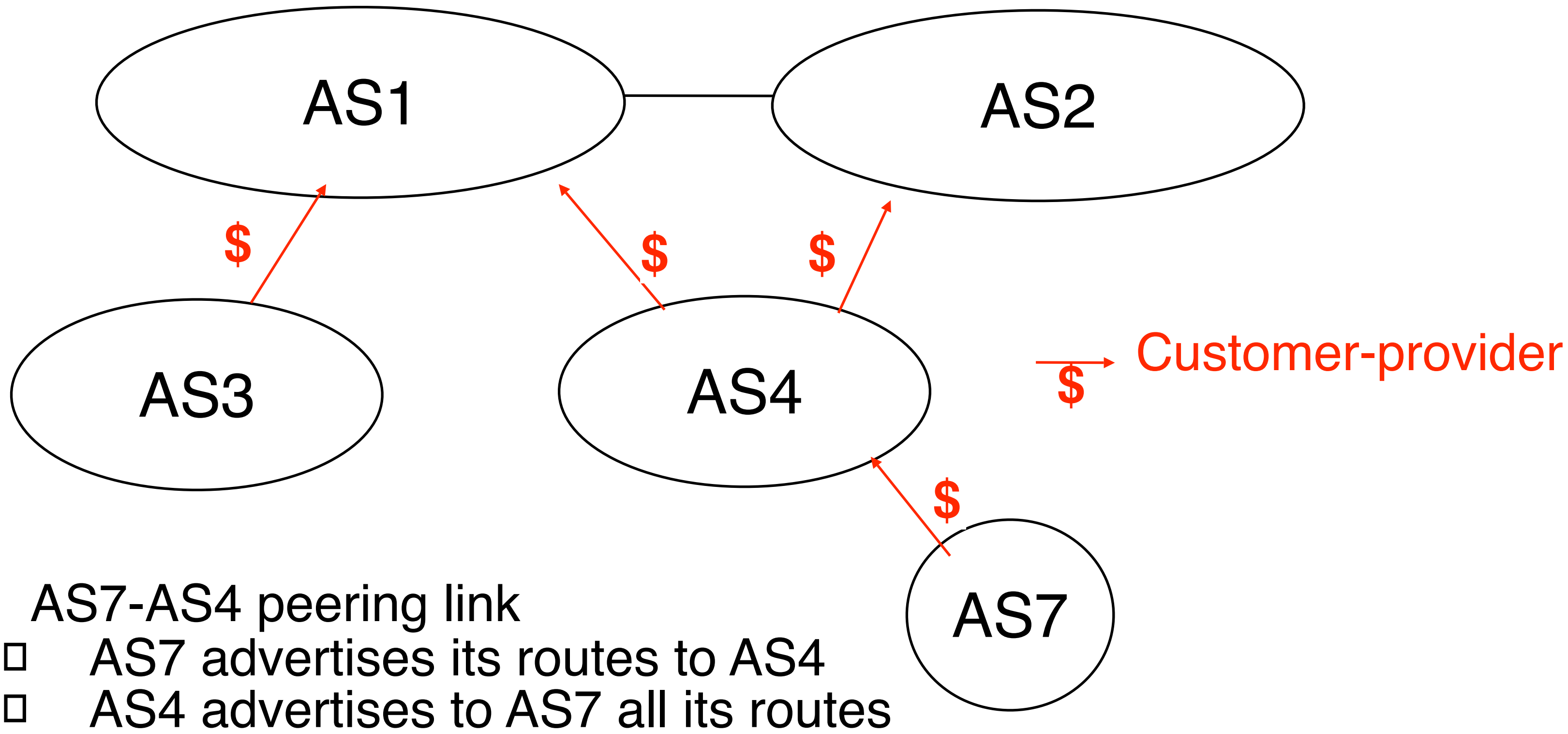
Shared-cost peering



- Principle
  - PeerX sends to PeerY its internal routes and the routes learned from its own customers
    - PeerY will use shared link to reach PeerX and PeerX's customers
    - PeerX's providers are not reachable via the shared link
  - PeerY sends to PeerX its internal routes and the routes learned from its own customers
    - PeerX will use shared link to reach PeerY and PeerY's customers
    - PeerY's providers are not reachable via the shared link

On link AS3-AS4
    AS3 advertises its internal routes
    AS4 advertises its internal routes and the routes learned from AS7 (its customer)
On link AS1-AS2
    AS1 advertises its internal routes and the routes received from AS3 and AS4 (its customers)
    AS2 advertises its internal routes and the routes learned from AS74(its customer)

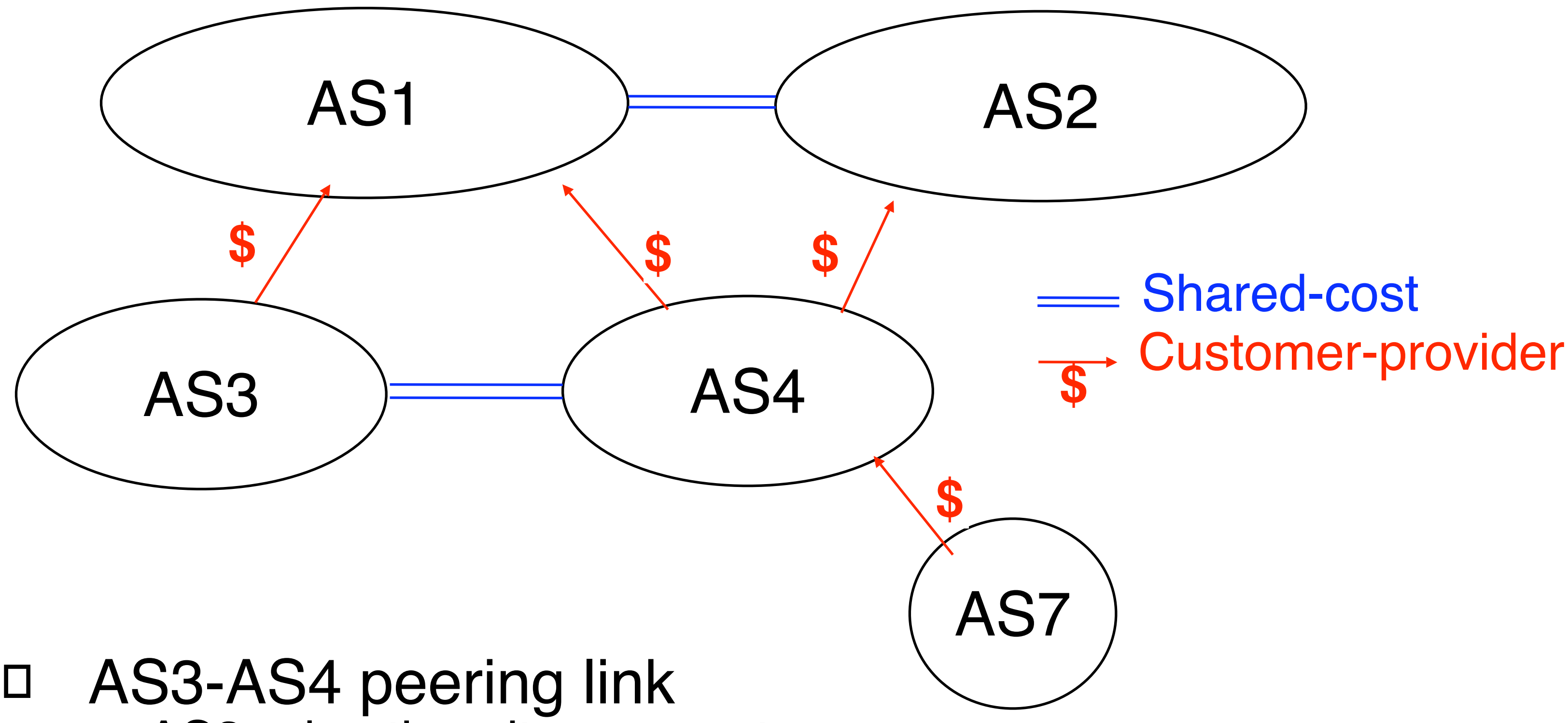
Customer-provider peering : example



- AS7-AS4 peering link
  - AS7 advertises its routes to AS4
  - AS4 advertises to AS7 all its routes

- AS4-AS2 peering link
  - AS4 advertises its own routes et those of its customers (AS7)
  - AS2 advertises to AS2 all known routes

Shared-cost peering : example



- AS3-AS4 peering link
  - AS3 advertises its own routes
  - AS4 advertises its own routes and those received from its clients (AS7)
- AS1-AS2 peering link
  - AS1 advertises its own routes and those received from its clients (AS3 and AS4)
  - AS1 advertises its own routes and those received from its clients (AS4)

Routing policies

- A domain specifies its routing policy by defining on each BGP router two sets of filters for each peer
  - Import filter
    - Specifies which routes can be accepted by the router among all the received routes from a given peer
  - Export filter
    - Specifies which routes can be advertised by the router to a given peer
- Filters can be defined in RPSL
  - Routing Policy Specification Language
    - defined in RFC2622 and examples in RFC2650
      - See also http://www.ripe.net/ripencc/pub-services/whois.html

RFC 2622 Routing Policy Specification Language (RPSL). C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, M. Terpstra. June 1999.

RFC 2650 Using RPSL in Practice. D. Meyer, J. Schmitz, C. Orange, M. Prior, C. Alaettinoglu. August 1999.

Internet Routing Registries contain the routing policies of various ISPs, see :

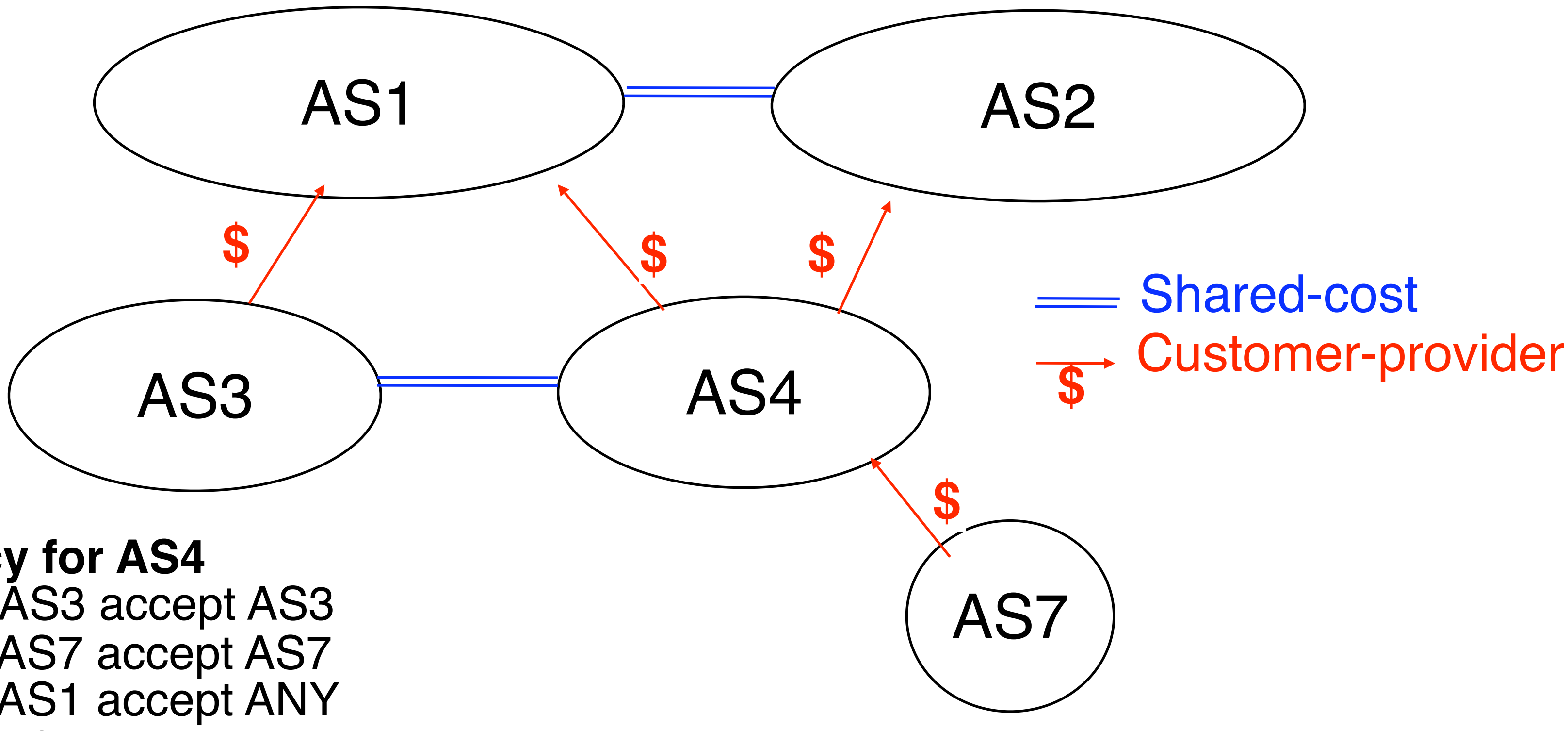
http://www.ripe.net/ripencc/pub-services/whois.html
http://www.arin.net/whois/index.html
http://www.apnic.net/apnic-bin/whois.pl

RPSL

- Simple import policies
  - Syntax
    - `import: from AS# accept list_of_AS`
  - Examples
    - `Import: from Belgacom accept Belgacom WIN`
    - `Import: from Provider accept ANY`

- Simple export policies
  - Syntax
    - `Export: to AS# announce list_of_AS`
  - Example
    - `Export: to Customer announce ANY`
    - `Export: to Peer announce Customer1 Customer2`

Routing policies
Simple example with RPSL



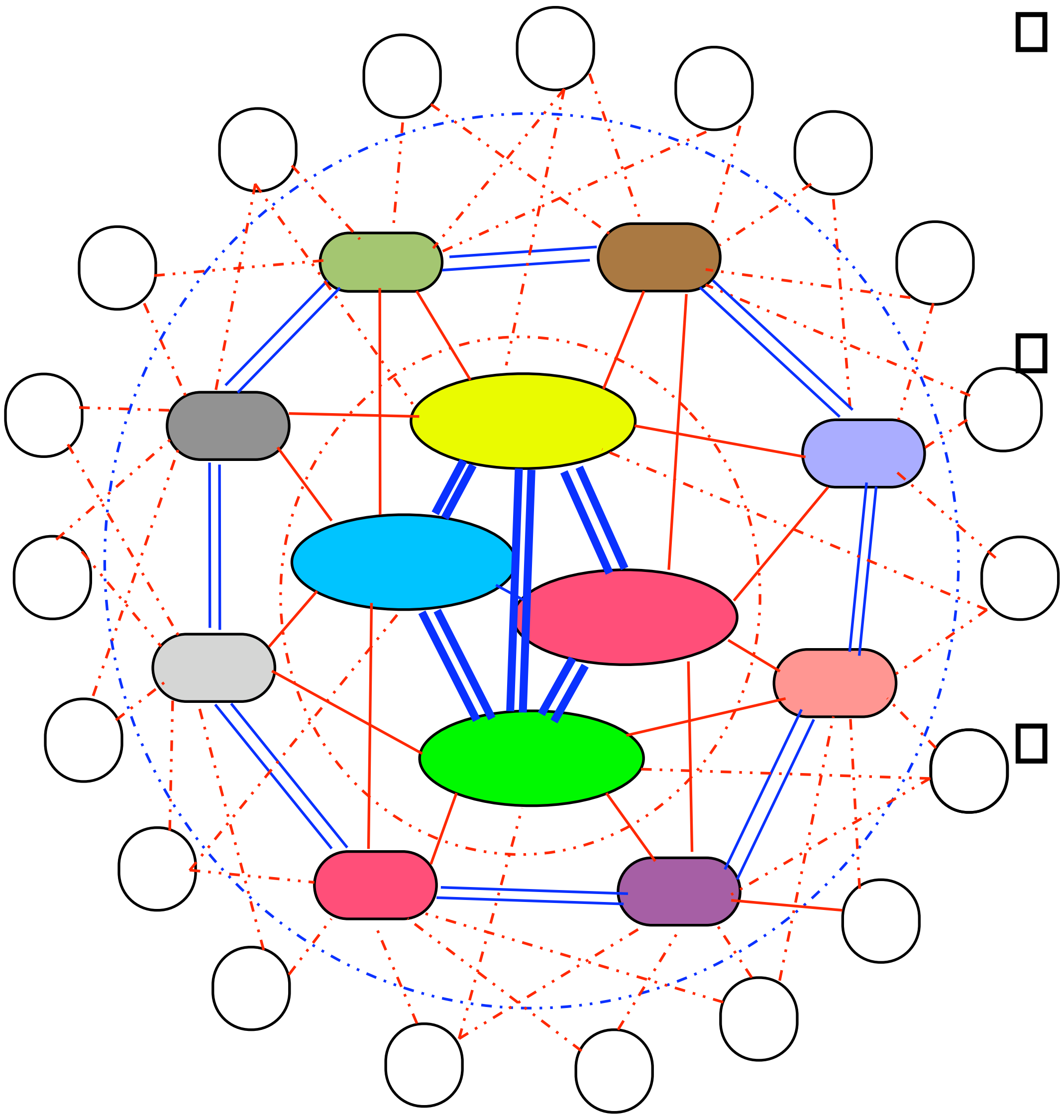
Import policy for AS4
Import: from AS3 accept AS3
import: from AS7 accept AS7
import: from AS1 accept ANY
import: from AS2 accept ANY

Export policy for AS4
export: to AS3 announce AS4 AS7
export: to AS7 announce ANY
export: to AS1 announce AS4 AS7
export: to AS2 announce AS4 AS7

Import policy for AS7
Import: from AS4 accept ANY

Export policy for AS4
export: to AS4 announce AS7

The organisation of the Internet



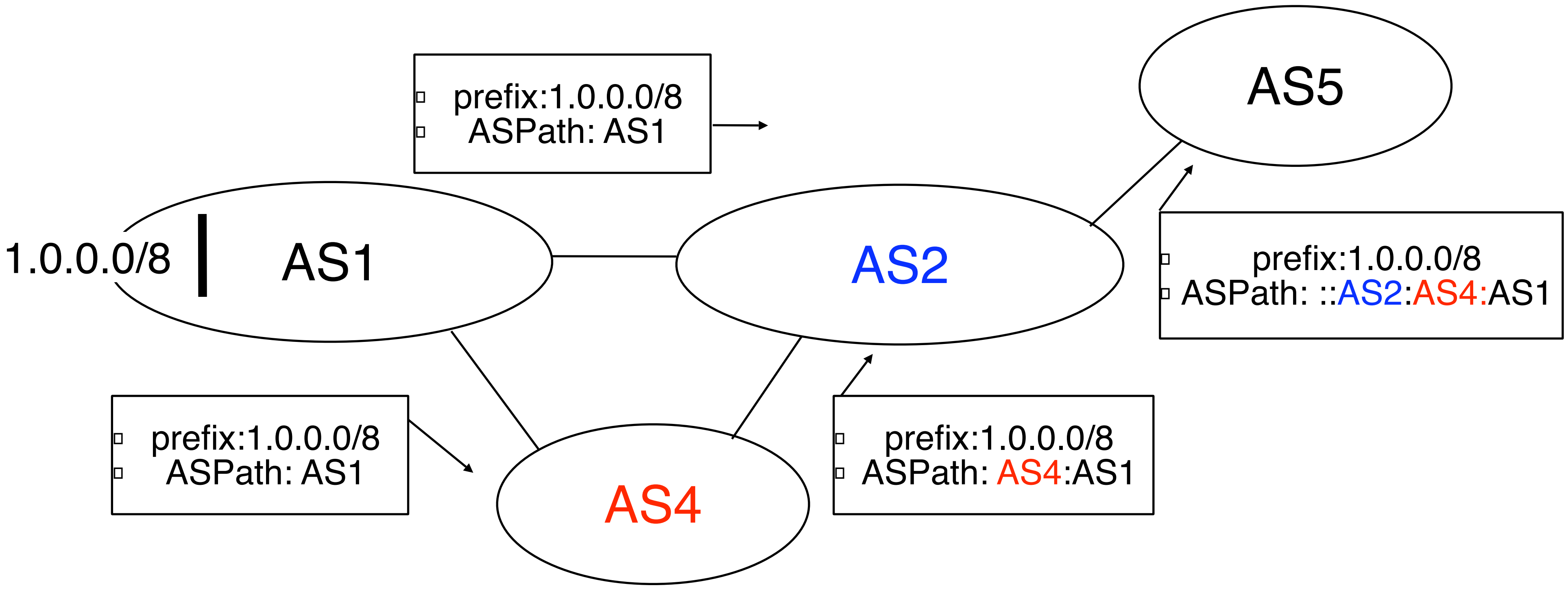
- Tier-1 ISPs
  - Dozen of large ISPs interconnected by shared-cost
  - Provide transit service
    - Uunet, Level3, OpenTransit, ...
- Tier-2 ISPs
  - Regional or National ISPs
  - Customer of T1 ISP(s)
  - Provider of T2 ISP(s)
  - shared-cost with other T2 ISPs
    - France Telecom, BT, Belgacom
- Tier-3 ISPs
  - Smaller ISPs, Corporate Networks, Content providers
  - Customers of T2 or T1 ISPs
  - shared-cost with other T3 ISPs

See :

L. Subramanian, S. Agarwal, J. Rexford, and RH Katz. Characterizing the Internet hierarchy from multiple vantage points. In IEEE INFOCOM, 2002

The Border Gateway Protocol

- Principle
  - Path vector protocol
    - BGP router advertises its best route to each destination

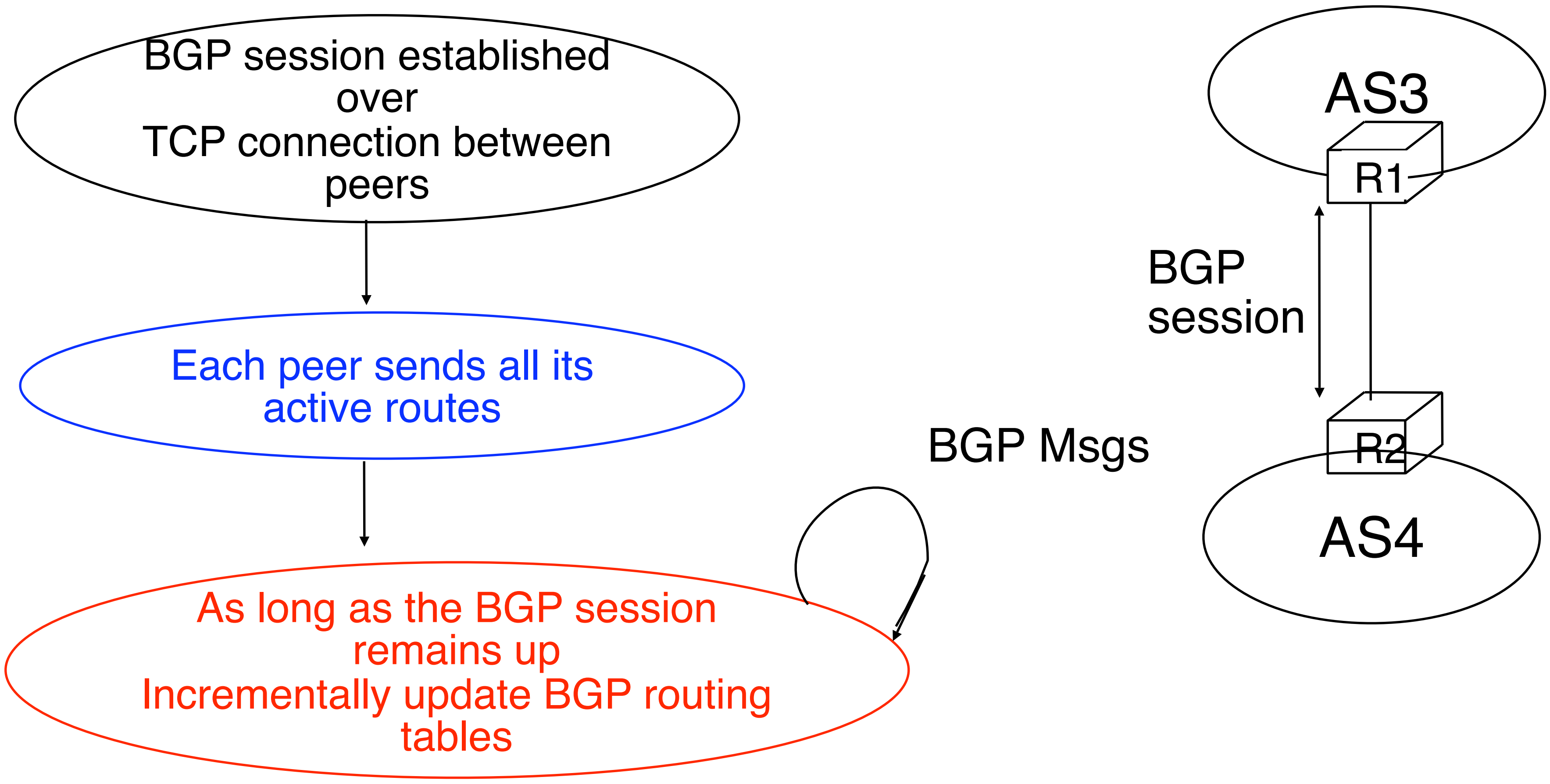


- ... with incremental updates
  - Advertisements are only sent when their content changes

BGP is defined in RFC4271

BGP : Principles of operation

- Principles
  - BGP relies on the incremental exchange of path vectors



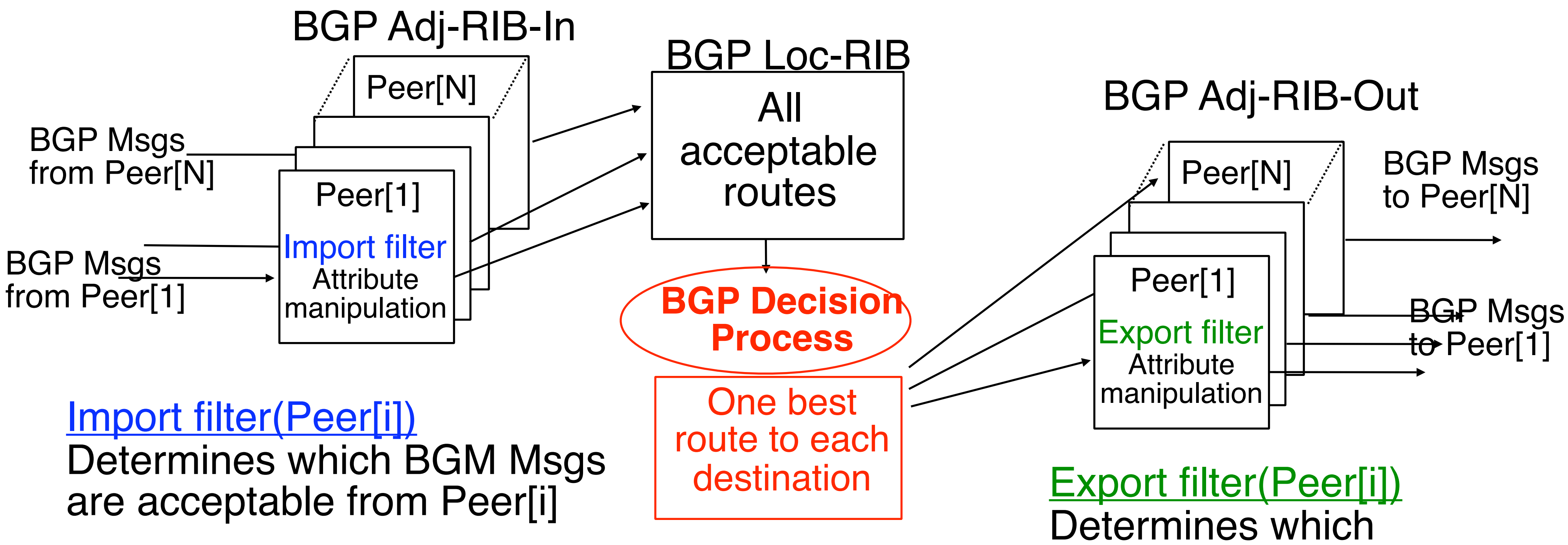
BGP : Principles of operation (2)

- Simplified model of BGP
  - 2 types of BGP path vectors

  - UPDATE
    - Used to announce a route towards one prefix
    - Content of UPDATE
      - Destination address/prefix
      - Interdomain path used to reach destination (AS-Path)
      - Nexthop (address of the router advertising the route)

  - WITHDRAW
    - Used to indicate that a previously announced route is not reachable anymore
    - Content of WITHDRAW
      - Unreachable destination address/prefix

Conceptual model of a BGP router



Import filter(Peer[i])
Determines which BGM Msgs
are acceptable from Peer[i]

Export filter(Peer[i])
Determines which
routes can be sent to Peer[i]

BGP Routing Information Base
Contains all the acceptable routes
learned from all Peers + internal routes

- BGP decision process selects *the* best route towards each destination

Where do the routes advertised by BGP routers come from ?

- Learned from another BGP router
  - Each BGP router advertises best route towards each destination
- Static route
  - Configured manually on the router
    - Ex : The BGP router at UCL advertises 130.104.0.0/16
    - Drawback
      - Requires manual configuration
    - Advantage
      - BGP advertisements are stable
- Learned from an intradomain routing protocol
  - BGP might try to aggregate the route before advertising it
  - Advantage :
    - BGP advertisements correspond to network status
  - Drawback
    - Routing instabilities inside a domain might propagate in Internet

BGP : Session Initialization

```
Initialize_BGP_Session(RemoteAS, RemoteIP)
{ /* Initialize and start BGP session */
/* Send BGP OPEN Message to RemoteIP on port 179*/
/* Follow BGP state machine */

/* advertise local routes and routes learned from peers*/
foreach (destination=d inside RIB)
 {
  B=build_BGP_UPDATE(d);
  S=apply_export_filter(RemoteAS,B);
  if (S<>NULL)
     { /* send UPDATE message */
       send_UPDATE(S,RemoteAS, RemoteIP)
     }
 }
/* entire RIB was sent */
/* new UPDATE will be sent only to reflect local or distant
   changes in routes */
...
}
```

Events during a BGP session

1. Addition of a new route to RIB
   - A new internal route was added on local router
     - static route added by configuration
     - Dynamic route learned from IGP
   - Reception of UPDATE message announcing a new or modified route
2. Removal of a route from RIB
   - Removal of an internal route
     - Static route is removed from router configuration
     - Intradomain route declared unreachable by IGP
   - Reception of WITHDRAW message
3. Loss of BGP session
   - All routes learned from this peer removed from RIB

Export and Import filters

```
BGPMsg Apply_export_filter(RemoteAS, BGPMsg)
{ /* check if Remote AS already received route */
if (RemoteAS isin BGPMsg.ASPath)
   BGPMsg==NULL;
/* Many additional export policies can be configured : */
/* Accept or refuse the BGPMsg */
/* Modify selected attributes inside BGPMsg */
}

BGPMsg apply_import_filter(RemoteAS, BGPMsg)
{ /* check that we are not already inside  ASPath */
 if (MyAS isin BGPMsg.ASPath)
   BGPMsg==NULL;
/* Many additional import policies can be configured : */
/* Accept or refuse the BGPMsg */
/* Modify selected attributes inside BGPMsg */
}
```


In the above export filter, we assume that the BGP sender does not send to PeerX the routes learned from this peer. This behavior is not required by the BGP specification, but is a common optimization, often called sender-side loop detection.

The check for the presence of the localAS number in the routes learned is specified in the BGP RFC.

BGP : Processing of UPDATES

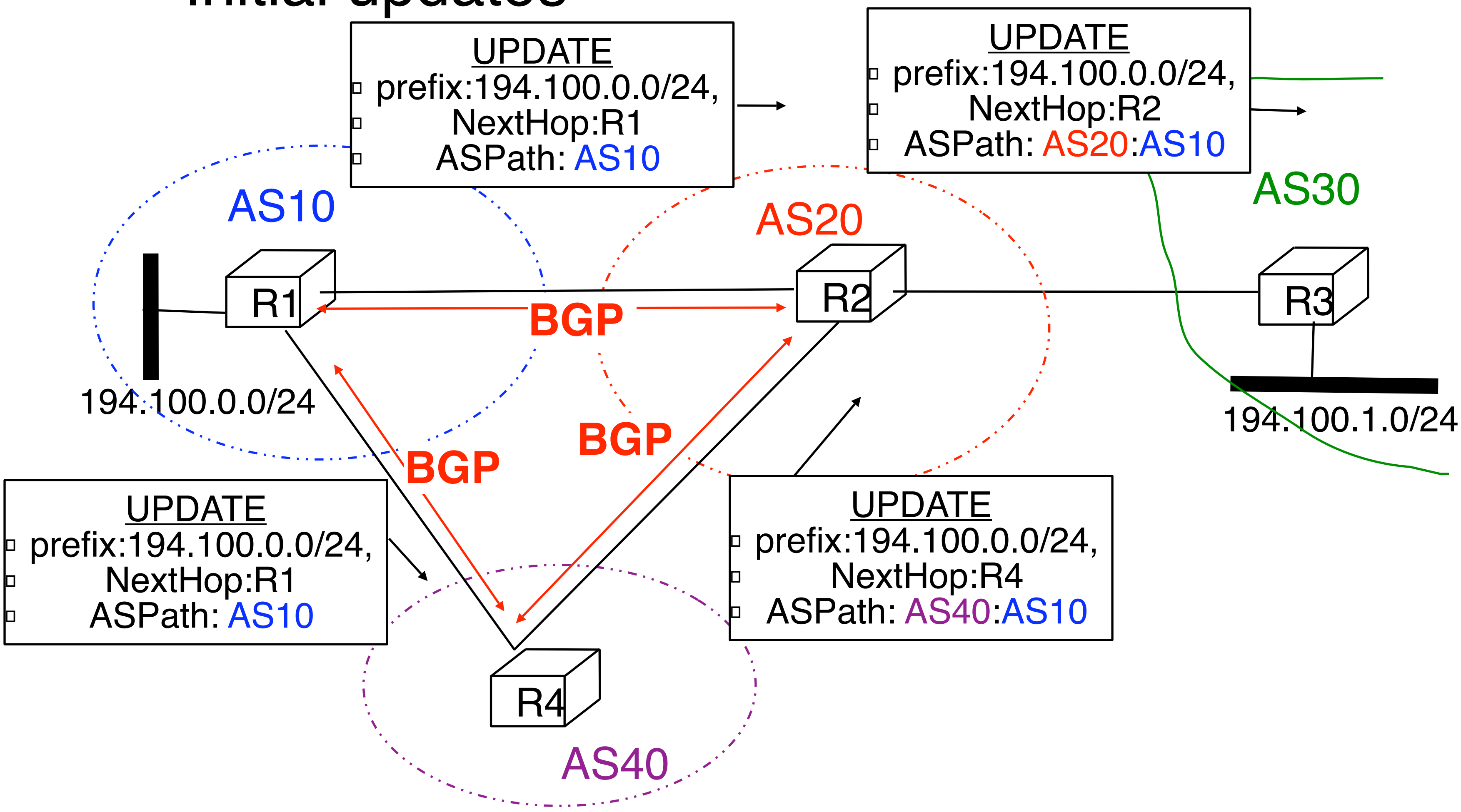
```
Recvd_BGPMsg(Msg, RemoteAS)
{
 B=apply_import_filer(Msg,RemoteAS);
 if (B==NULL) /* Msg not acceptable */
     exit();
 if IsUPDATE(Msg)
 {
  Old_Route=BestRoute(Msg.prefix);
  Insert_in_RIB(Msg);
  Run_Decision_Process(RIB);
  if (BestRoute(Msg.prefix)<>Old_Route)
  { /* best route changed */
    B=build_BGP_Message(Msg.prefix);
    S=apply_export_filter(RemoteAS,B);
    if (S<>NULL) /* announce best route */
     send_UPDATE(S,RemoteAS);
    else if (Old_Route<>NULL)
     send_WITHDRAW(Msg.prefix);
 } ...
```

BGP : Processing of WITHDRAW

```
Recvd_Msg(Msg, RemoteAS)
...
if IsWITHDRAW(Msg)
 {
  Old_Route=BestRoute(Msg.prefix);
  Remove_from_RIB(Msg);
  Run_Decision_Process(RIB);
  if (Best_Route(Msg.prefix)<>Old_Route)
  { /* best route changed */
    B=build_BGP_Message(d);
    S=apply_export_filter(RemoteAS,B);
    if (S<>NULL) /* still one best route */
       send_UPDATE(S,RemoteAS, RemoteIP);
    else if(Old_Route<>NULL)/* no best route anymore */
       send_WITHDRAW(Msg.prefix,RemoteAS,RemoteIP);
  }
 }
}
```

BGP and IP
A first example

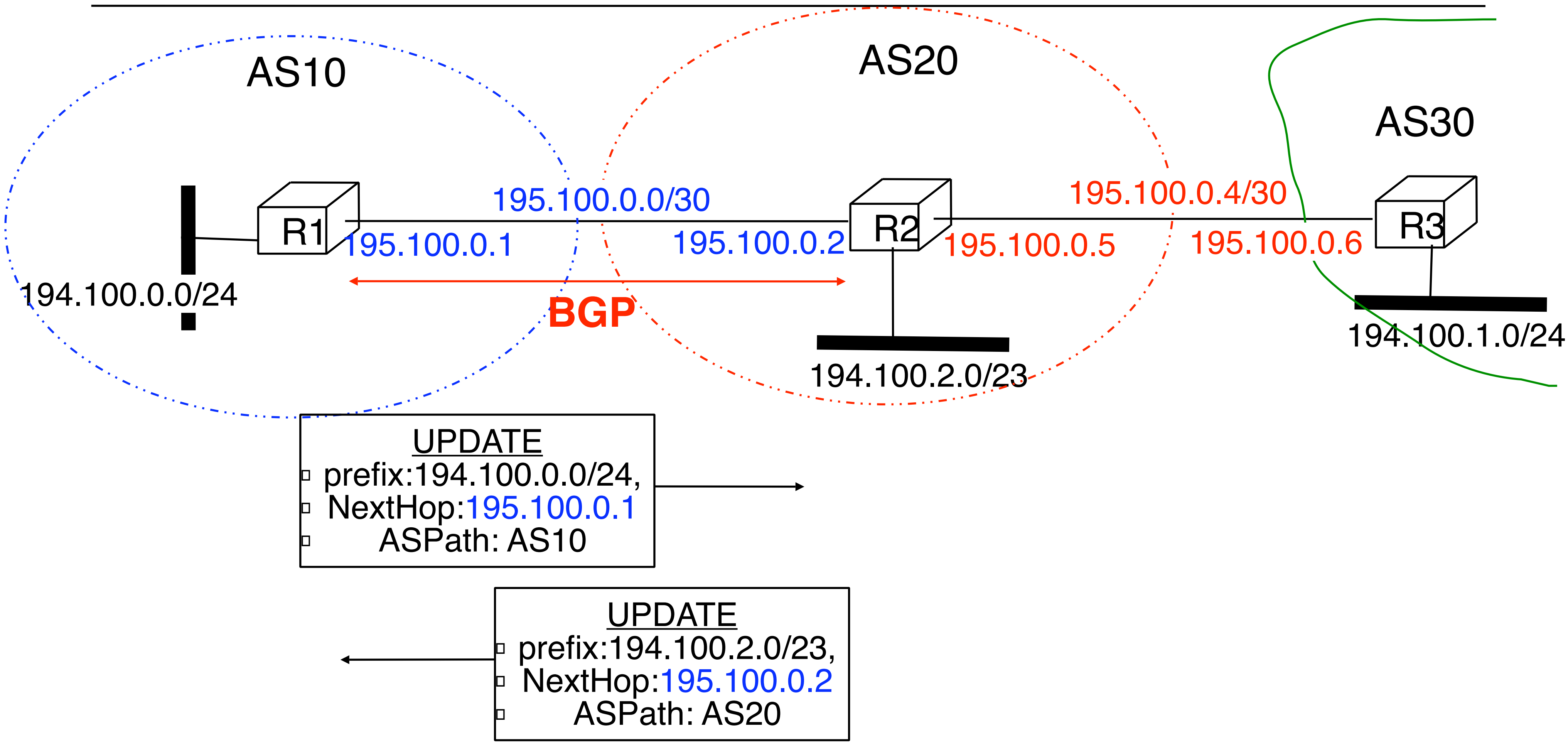
- Initial updates



- What happens if link AS10-AS20 goes down ?

If link AS10-AS20 goes down, AS20 will not consider anymore the path learned from AS10. It will thus remove this path from its routing table and will instead select the path learned from AS40. This will force AS20 to send the following UPDATE to AS30 :

BGP and IP
A second example



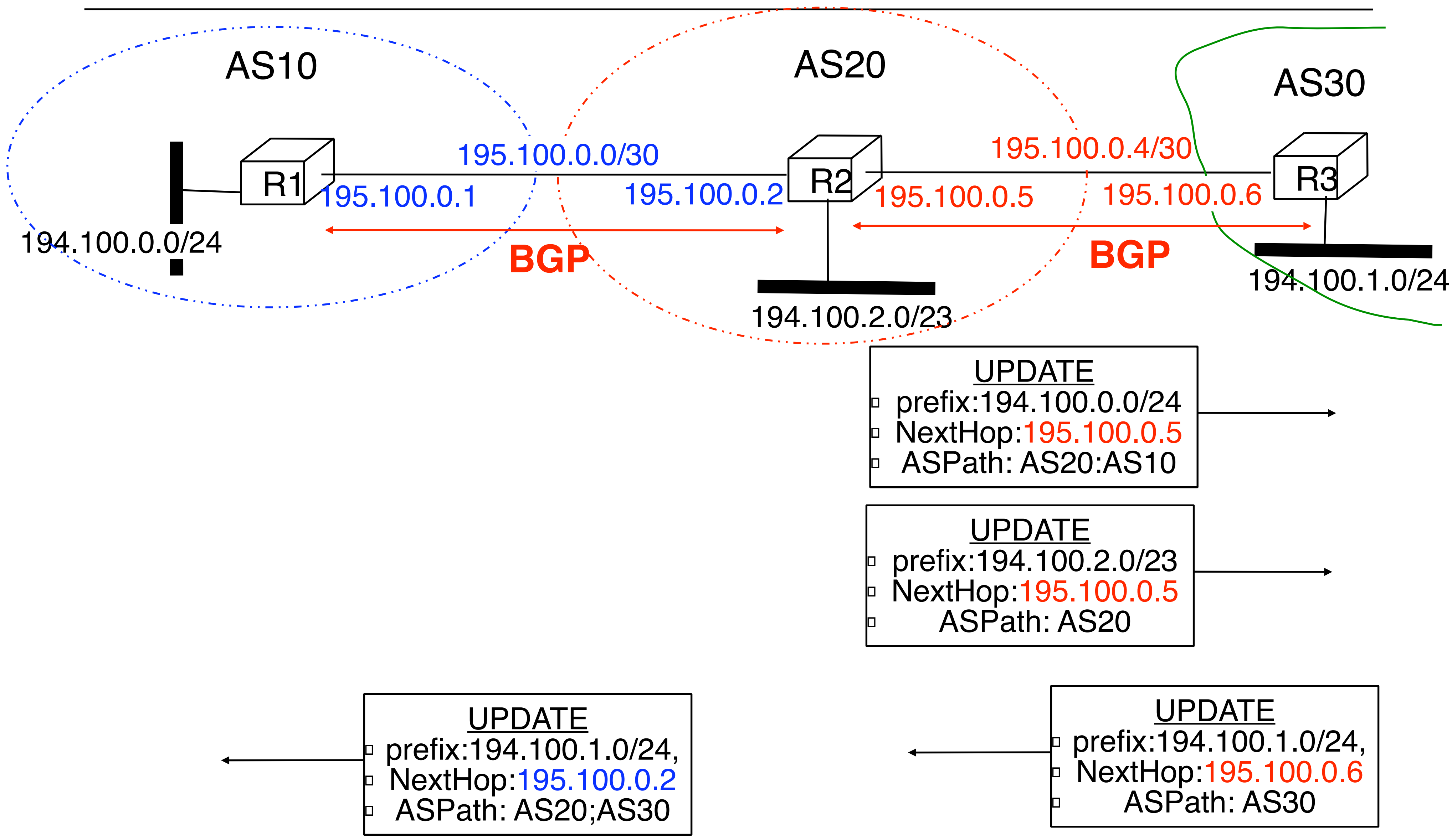
- Main Path attributes of UPDATE message
  - NextHop : IP address of router used to reach destination
  - ASPath : Path followed by the route advertisement

In this example, we only consider the BGP messages concerning the following IP networks :194.100.0.0/24, 194.100.1.0.0/24 and 194.100.2.0/23. Routes concerning networks 195.100.* also need to be distributed in practice, but they are not considered in the example.

The UPDATE message carries the ASPath in order to be able to detect routing loops.

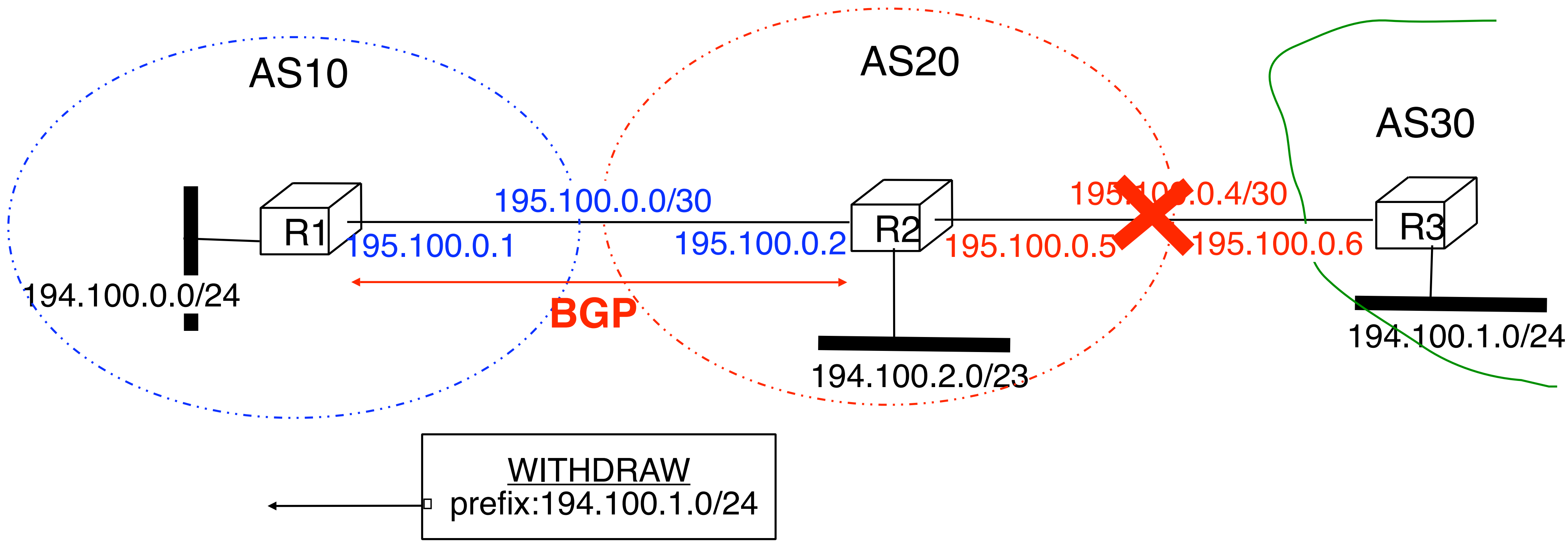
The nexthop information in the UPDATE is often equal to the IP address of the router advertising the route, but it can be sometimes useful to advertise as a next hop another IP address than the address of the router producing the BGP UPDATE message. For example, a router supporting BGP could advertise a route on behalf of another router who cannot run the BGP protocol.

BGP and IP
A second example (2)



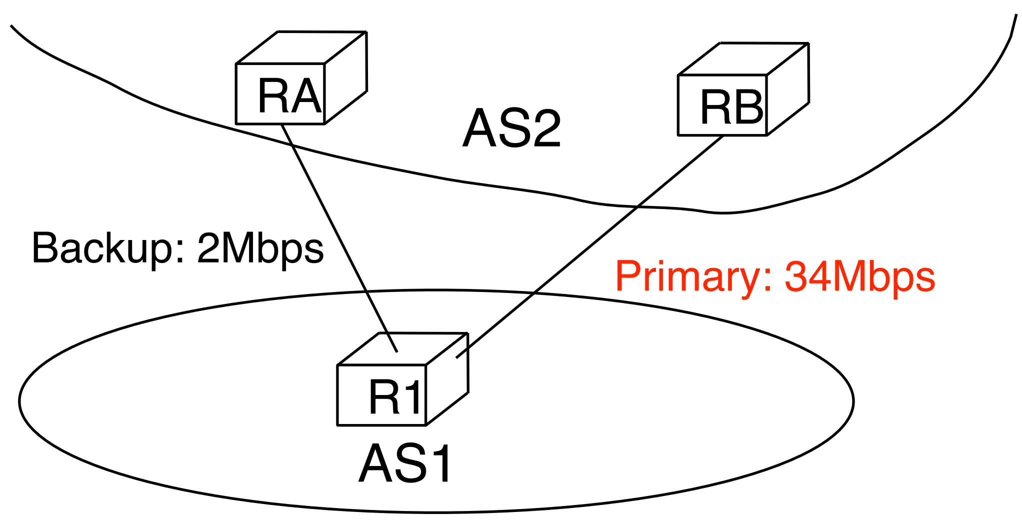
In this example, we only consider the BGP messages concerning the following IP networks :194.100.0.0/24, 194.100.1.0.0/24 and 194.100.2.0/23. Routes concerning networks 195.100.* also need to be distributed, but they are not considered in the example.

BGP and IP
A second example (3)

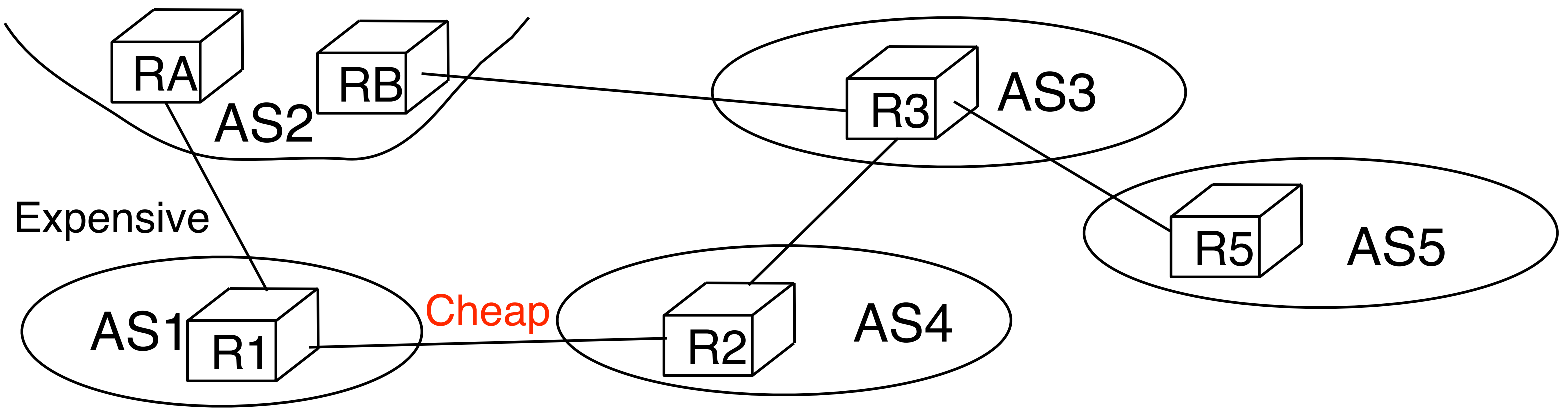


In this example, we only consider the BGP messages concerning the following IP networks :194.100.0.0/24, 194.100.1.0/24 and 194.100.2.0/23. Routes concerning networks 195.100.* also need to be distributed, but they are not considered in the example.

How to prefer some routes over others ?

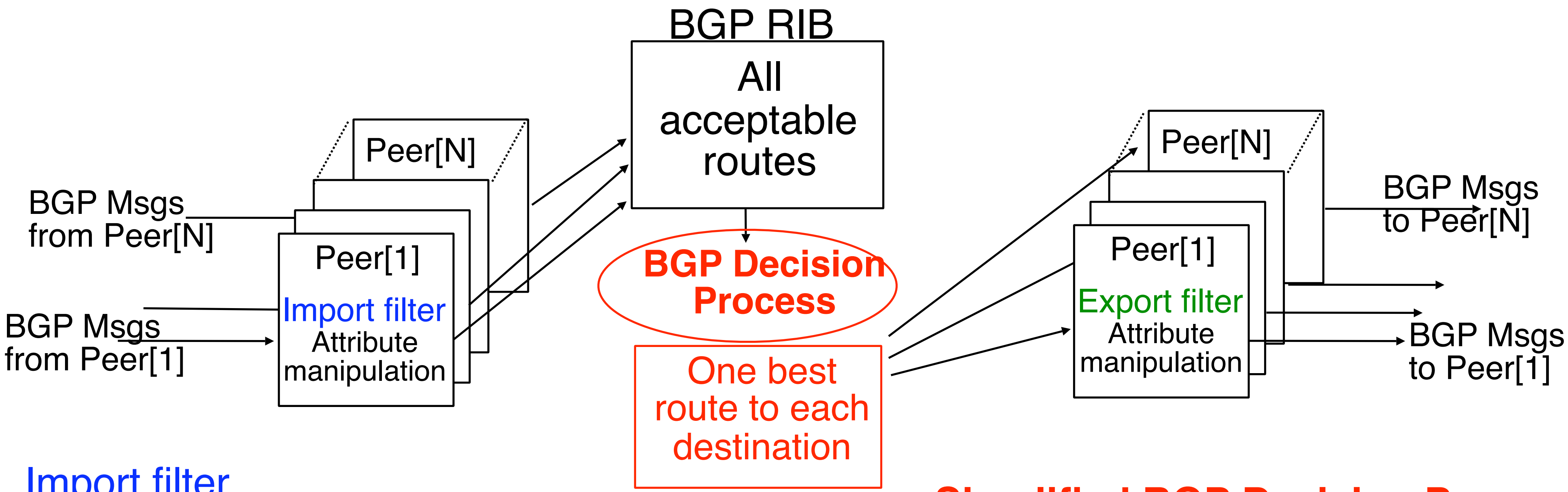


- How to ensure that packets will flow on primary link ?



- How to prefer cheap link over expensive link ?

How to prefer some routes over others (2) ?



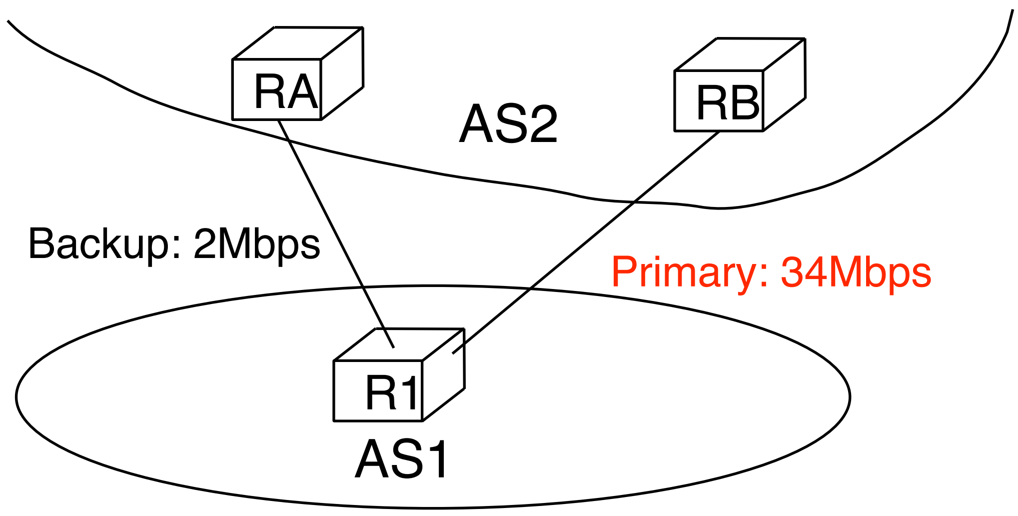
Import filter

- Selection of acceptable routes
- Addition of `local-pref` attribute inside received BGP Msg
  - Normal quality route : `local-pref`=100
  - Better than normal route :`local-pref`=200
  - Worse than normal route :`local-pref`=50

Simplified BGP Decision Process

- Select routes with highest `local-pref`
- If there are several routes, choose routes with the shortest ASPath
- If there are still several routes tie-breaking rule

How to prefer some routes over others (3) ?

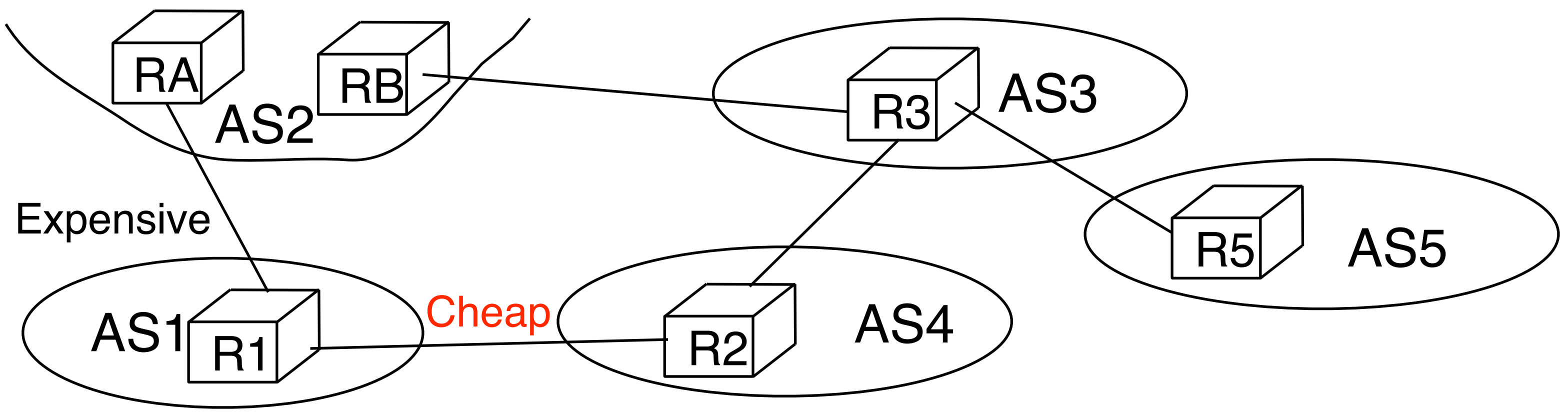


RPSL-like policy for AS1
aut-num: AS1
import: from AS2 RA at R1 set localpref=100;
from AS2 RB at R1 set localpref=200;
accept ANY
export: to AS2 RA at R1 announce AS1
to AS2 RB at R1 announce AS1

RPSL-like policy for AS2
aut-num: AS2
import: from AS1 R1 at RA set localpref=100;
from AS1 R1 at RB set localpref=200;
accept AS1
export: to AS1 R1 at RA announce ANY
to AS2 R1 at RB announce ANY

Note that in RPSL, the set localpref construct does not exist. It is replaced with action preference=x. Unfortunately, in RPSL the routes with the lowest preference are preferred. RPSL uses thus the opposite of local-pref....

How to prefer some routes over others (4) ?

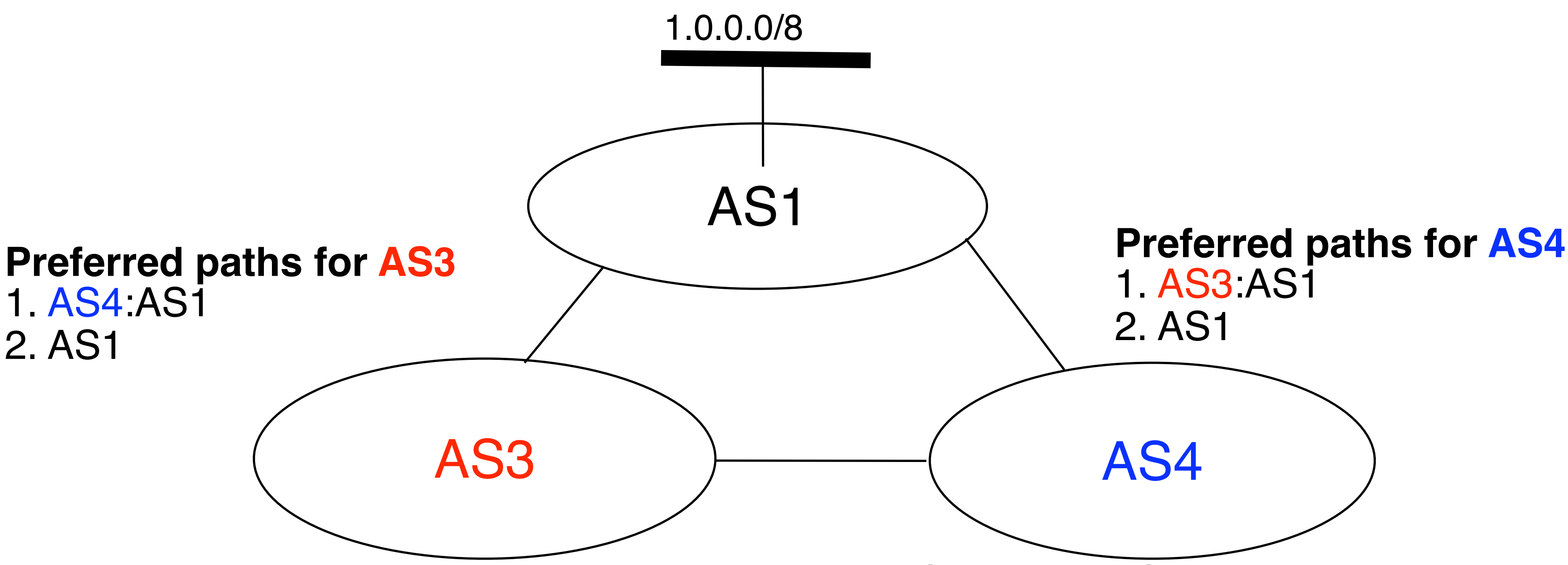


RPSL policy for AS1
aut-num: AS1
import: from AS2 RA at R1 set localpref=100;
from AS4 R2 at R1 set localpref=200;
accept ANY
export: to AS2 RA at R1 announce AS1
to AS4 R2 at R1 announce AS1

- AS1 will prefer to send packets over the cheap link
- But the flow of the packets destined to AS1 will depend on the routing policy of the other domains

Limitations of `local-pref`

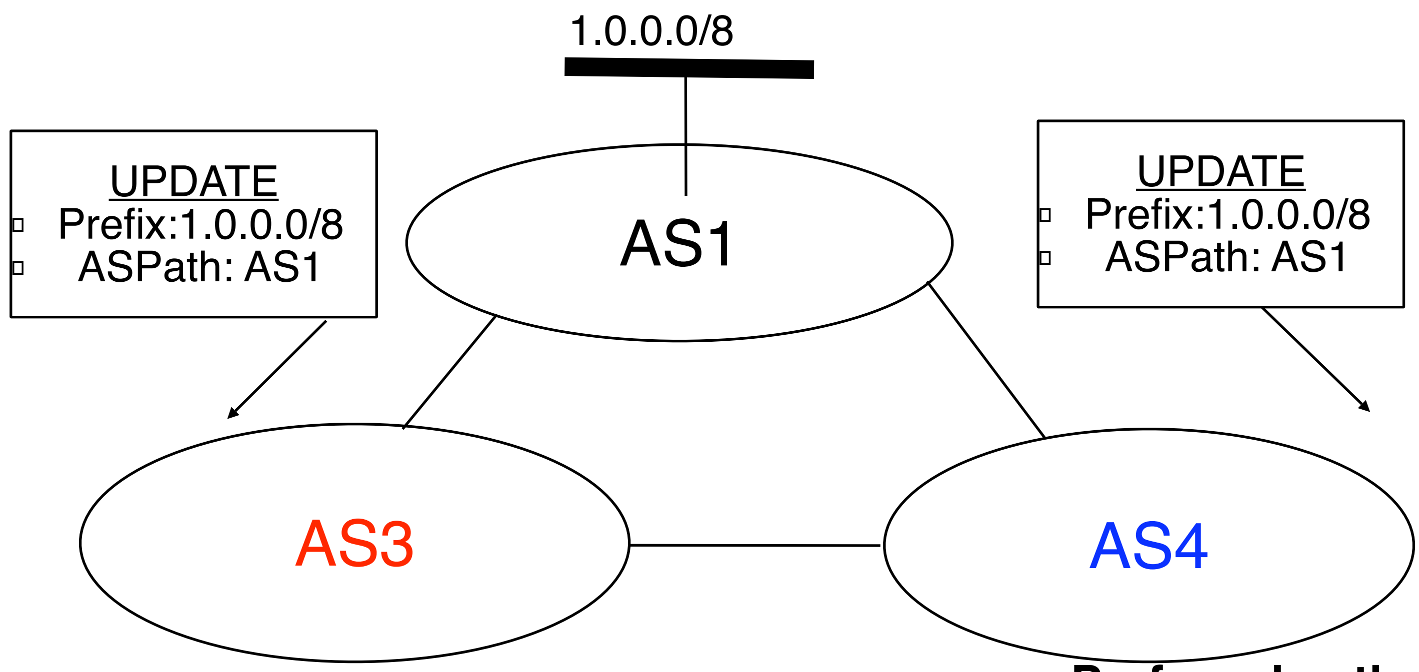
- In theory
  - Each domain is free to define its order of preference for the routes learned from external peers



  - How to reach 1.0.0.0/8 from AS3 and AS4 ?

Limitations of `local-pref` (2)

- AS1 sends its UPDATE messages ...



Preferred paths for AS3
1. AS4:AS1
2. AS1

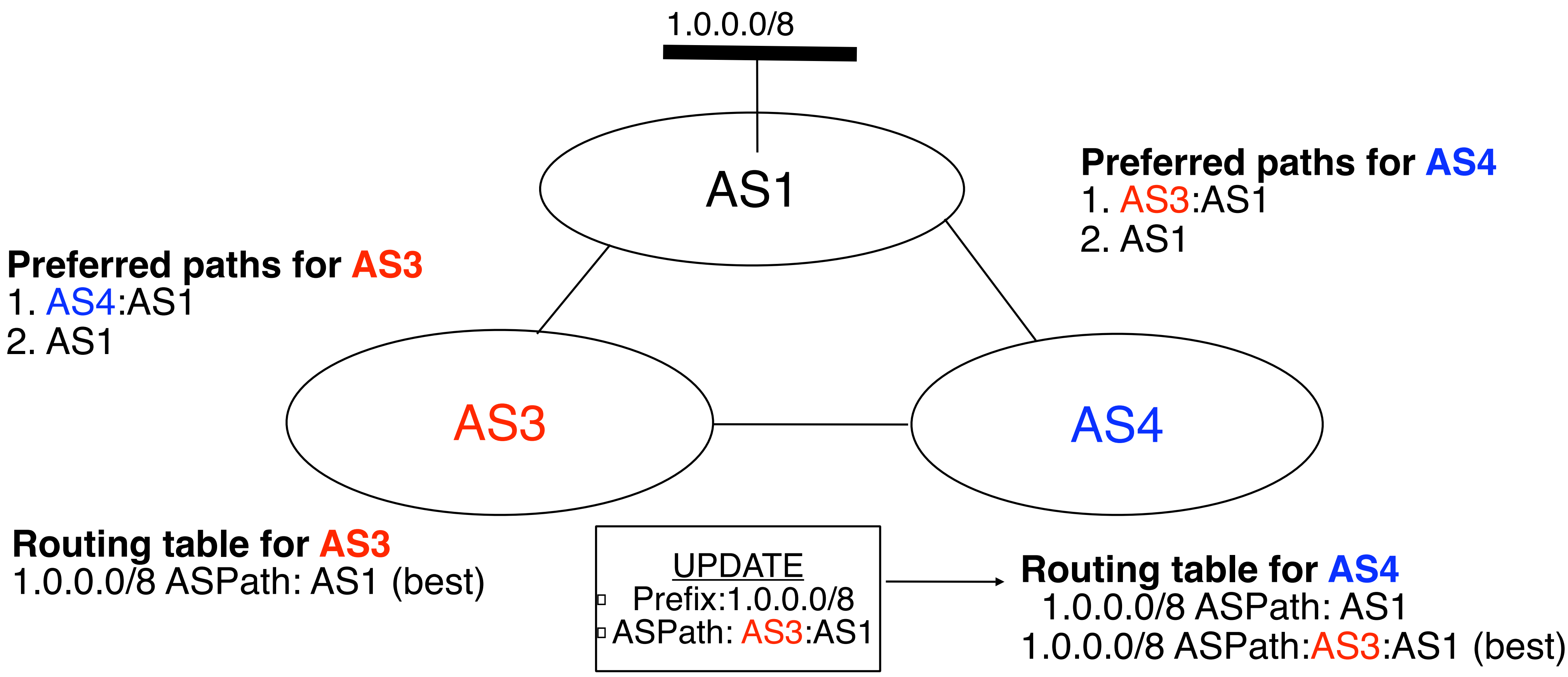
Preferred paths for AS4
1. AS3:AS1
2. AS1

Routing table for AS3
1.0.0.0/8 ASPath: AS1 (best)

Routing table for AS4
1.0.0.0/8 ASPath: AS1 (best)

Limitations of `local-pref` (3)

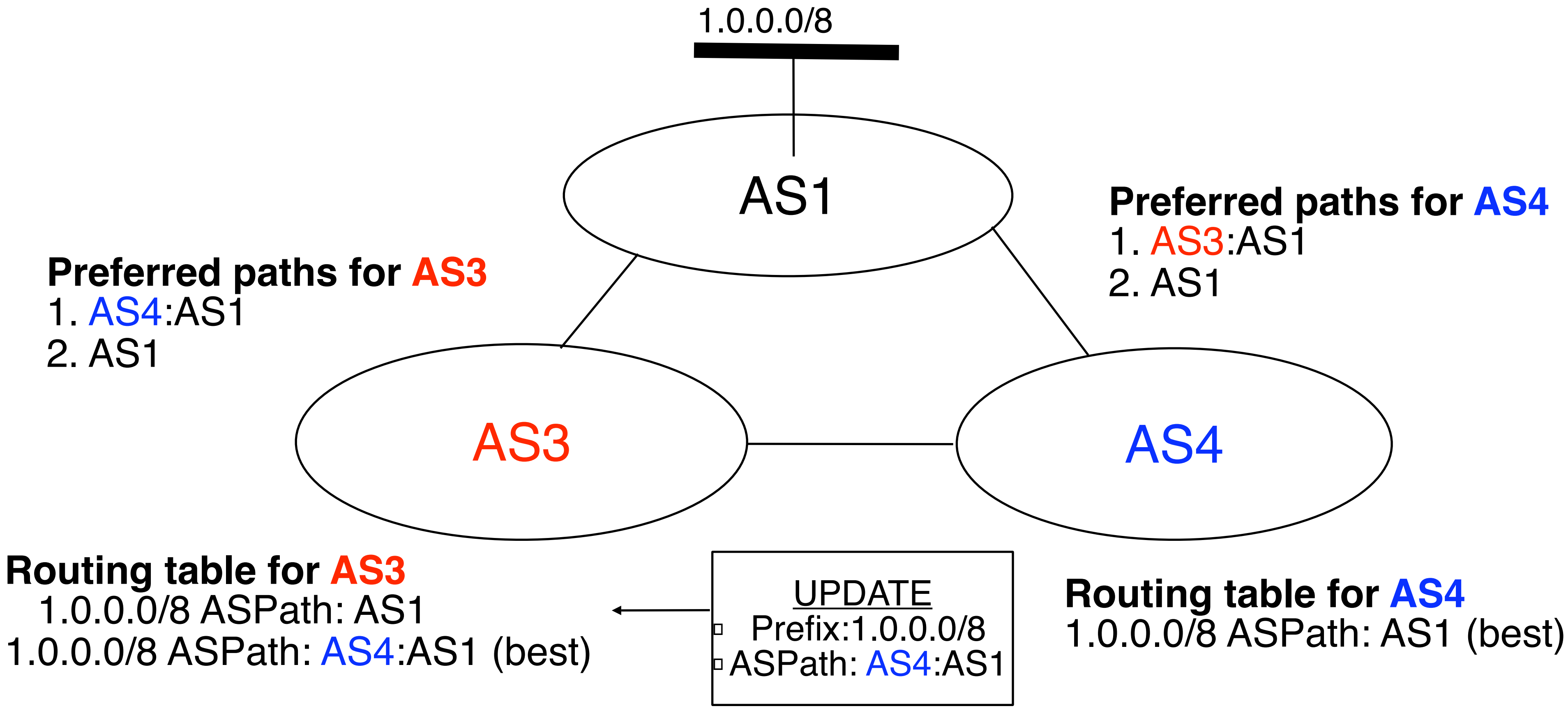
- First possibility
  - AS3 sends its UPDATE first...



- Stable route assignment

Limitations of `local-pref` (4)

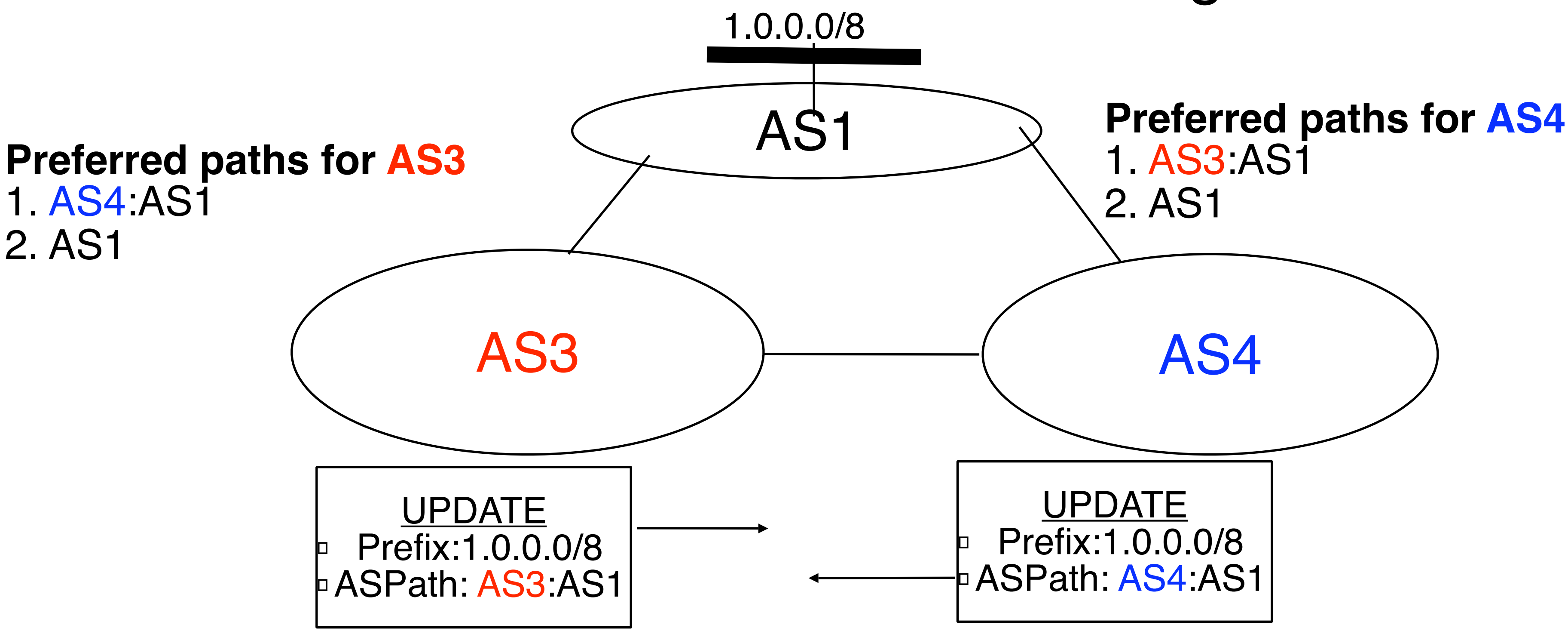
- Second possibility
  - AS4 sends its UPDATE first...



- Another (but different) stable route assignment

Limitations of `local-pref` (5)

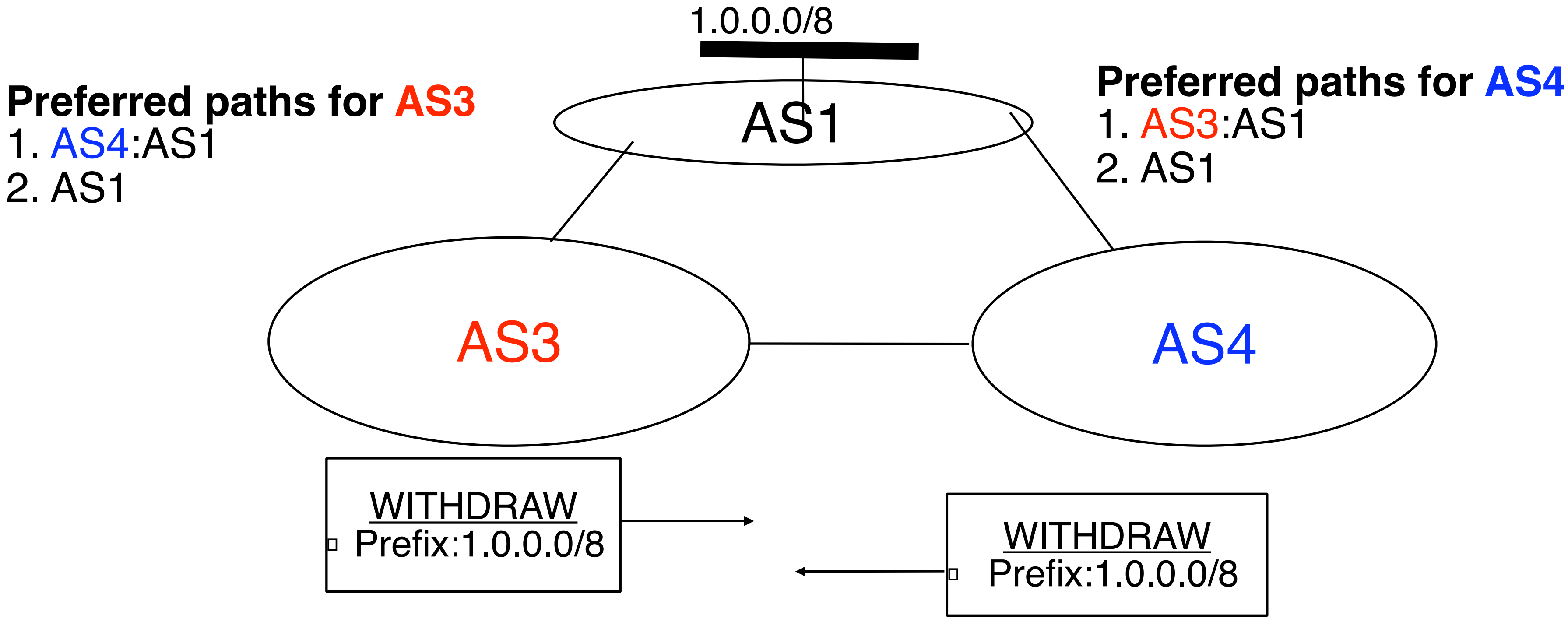
- Third possibility
  - AS3 and AS4 send their UPDATE together...



- AS3 prefers the indirect path and will thus send withdraw since the chosen best path is via AS4
- AS4 prefers the indirect path and will thus send withdraw since the chosen best path is via AS3

Limitations of `local-pref` (6)

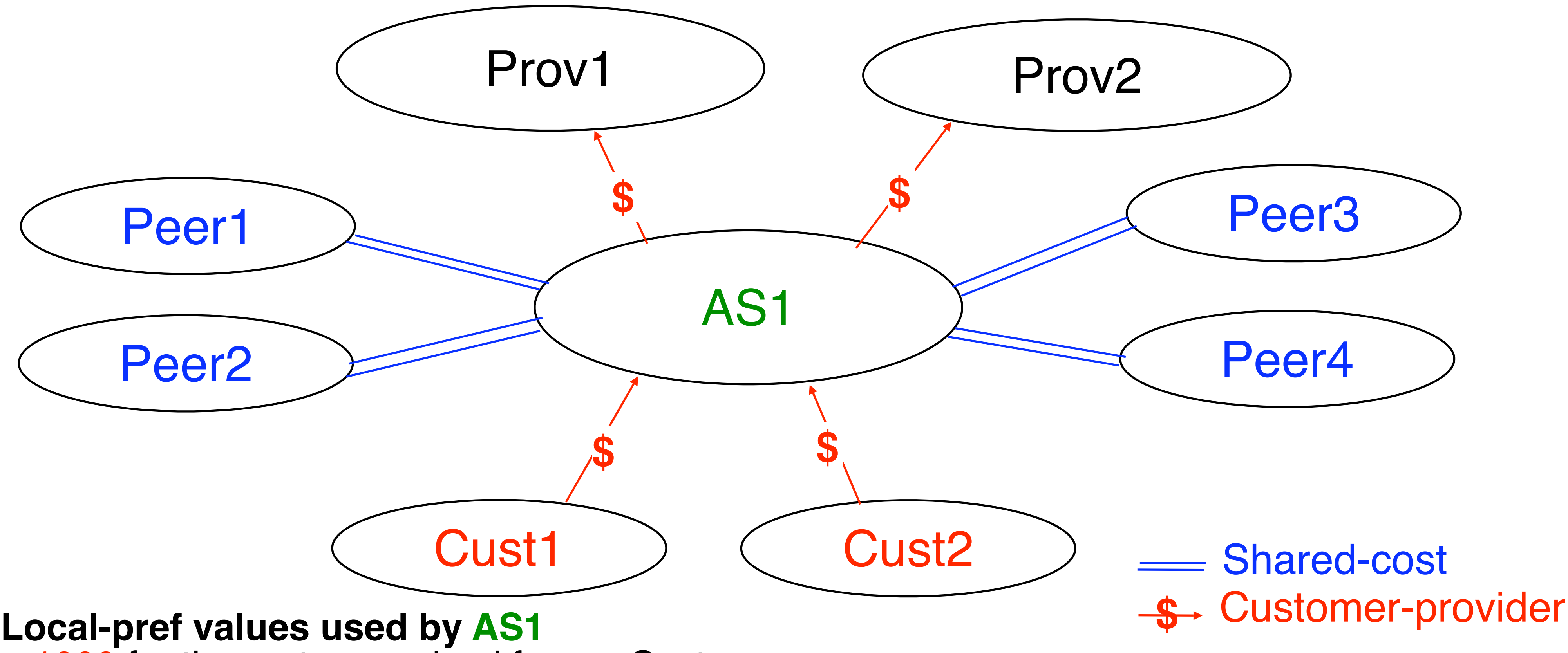
- Third possibility (cont.)
  - AS3 and AS4 send their UPDATE together...



- AS3 learns that the indirect route is not available anymore
  - AS3 will reannounce its direct route...
- AS4 learns that the indirect route is not available anymore
  - AS4 will reannounce its direct route...

`local-pref` and economical relationships

- In practice, `local-pref` is often used to enforce economical relationships



Local-pref values used by AS1
\> 1000 for the routes received from a Customer
500 – 999 for the routes learned from a Peer
< 500 for the routes learned from a Provider

This local-pref settings corresponds to the economical relationships between the various ASes.
Since AS1 is paid to carry packets towards Cust1 and Cust2, it will select a route towards those networks whenever possible.
Since AS1 does not need to pay to carry packets towards Peer1-4, AS1 will select a route towards those networks whenever possible.
AS1 will only utilize the routes receive from its providers when there is no other choice.

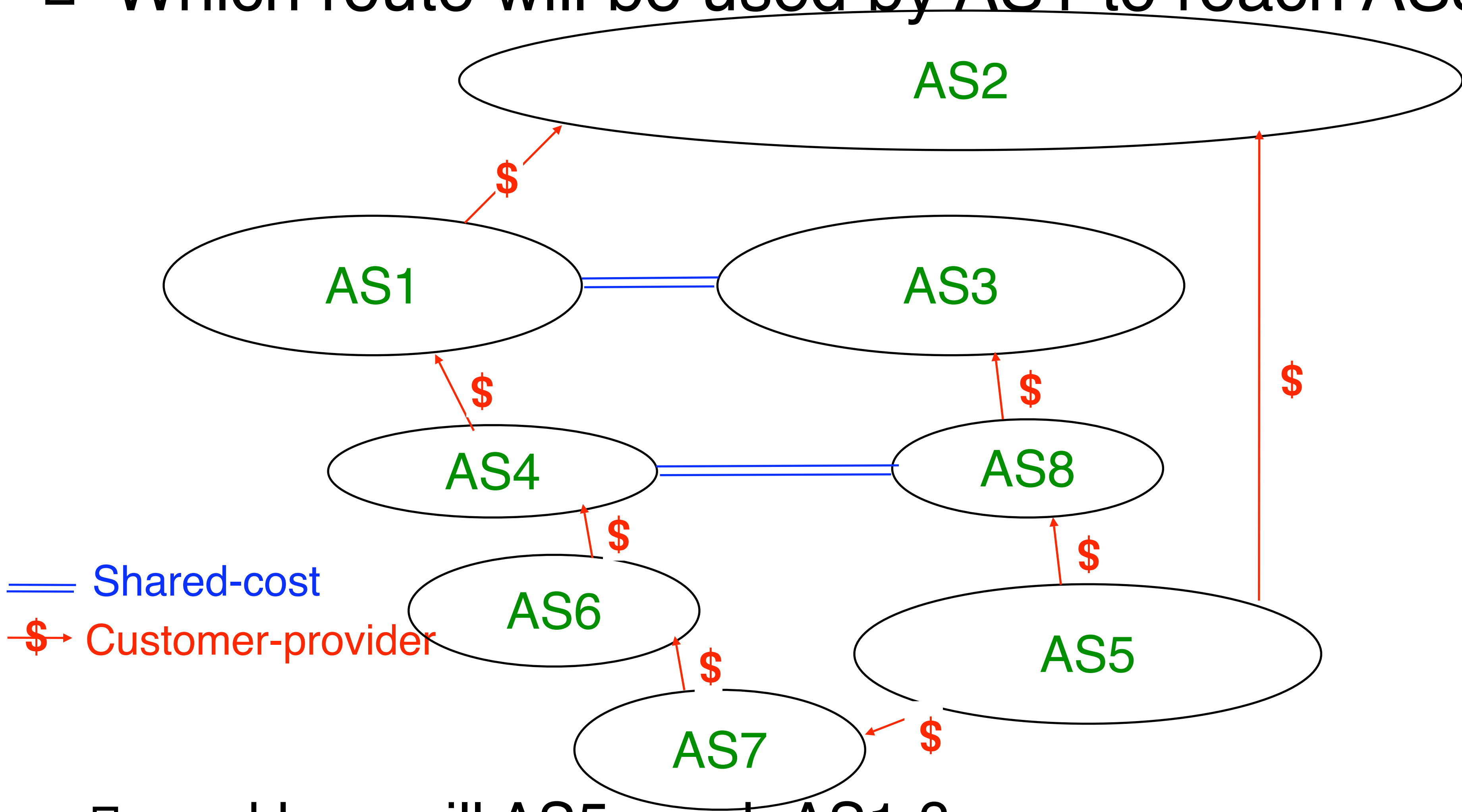
It is shown in the following papers that this way of utilizing the local-pref attribute leads to stable BGP routes :
Lixin Gao, Timothy G. Griffin, and Jennifer Rexford, "Inherently safe backup routing with BGP," Proc. IEEE INFOCOM, April 2001
Lixin Gao and Jennifer Rexford, "Stable Internet routing without global coordination," IEEE/ACM Transactions on Networking, December 2001, pp. 681-692

The RPSL policy of AS1 could be as follows :
RPSL policy for AS1

```
aut-num: AS1
import:    from  Cust1 action set localpref=200; accept Cust1
           from  Cust2 action set localpref=200; accept Cust2
           from  Peer1 action set localpref=150; accept Peer1
           from  Peer2 action set localpref=160; accept Peer2
           from  Peer3 action set localpref=170; accept Peer3
           from  Peer4 action set localpref=180; accept Peer4
           from  Prov1 action set localpref=100; accept ANY
           from  Prov2 action set localpref=100; accept ANY
```

Consequence of this utilisation of `local-pref`

- Which route will be used by AS1 to reach AS5 ?



- and how will AS5 reach AS1 ?

 Internet paths are often asymmetrical

Due to the utilization of the local-pref attribute, some paths on the Internet are longer than their optimum length, see :

Lixin Gao and Feng Wang , The Extent of AS Path Inflation by Routing Policies, GlobalInternet 2002

Security issues with interdomain routing

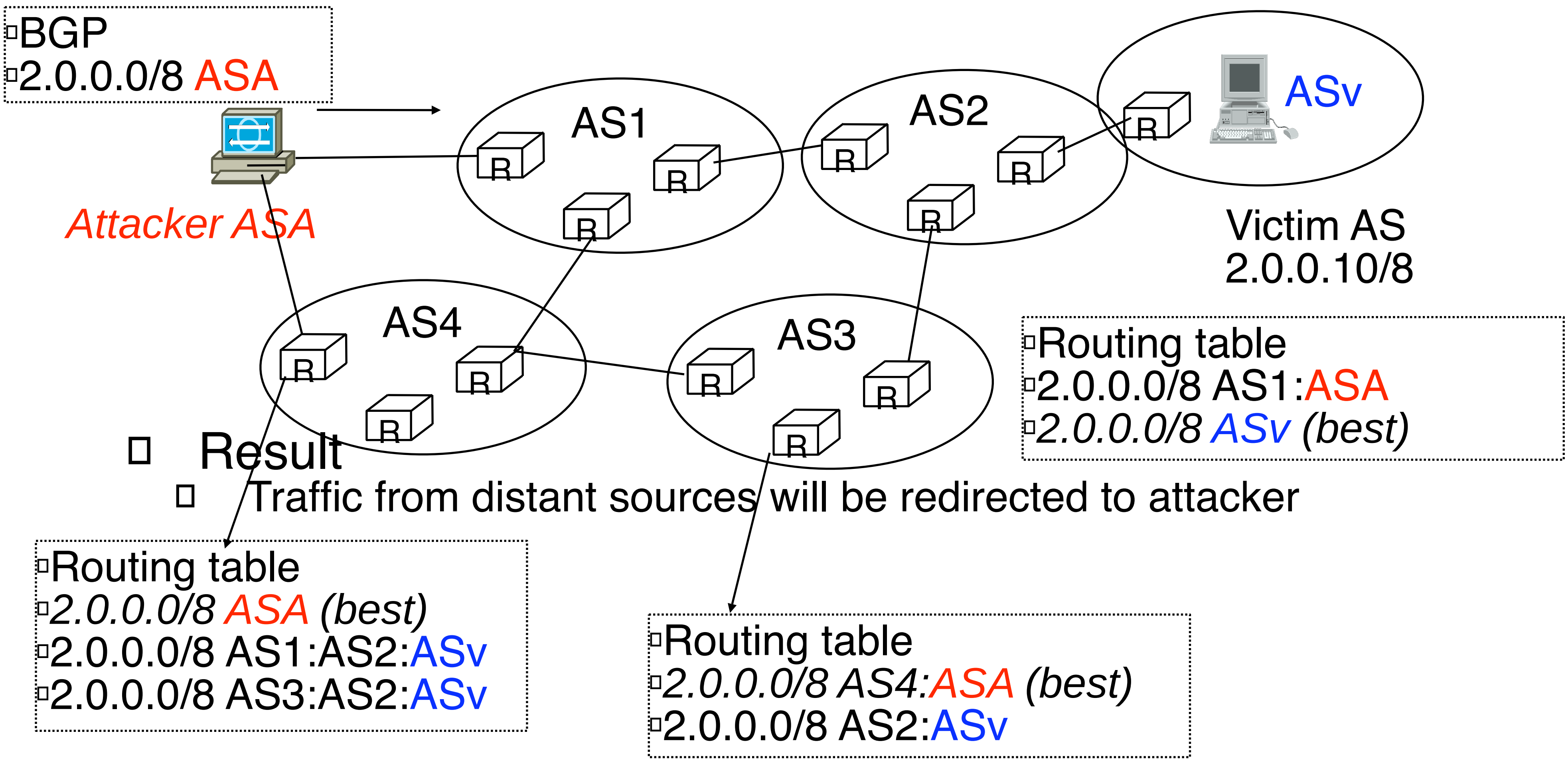
- Major issue
  - A BGP router from an AS should only advertise legitimate prefixes
    - Unfortunately, BGP does not contain a mechanism to prove that a route is legitimate
    - Configuration errors, intentional or not, are common
  - Possible consequence
    - Risk of traffic redirection / MITM
      - an attacker AS could advertise the prefix of a large bank or e-commerce site and redirect packets to his own site
    - Risk of Denial of service through blackholing
      - an attacker AS could advertise the prefix and drop all received packets

For a discussion of BGP security problems, see e.g.

S. Murphy, BGP Security Vulnerabilities Analysis, Internet draft, draft-murphy-bgp-vuln-02.txt, March 2003

Invalid advertisements

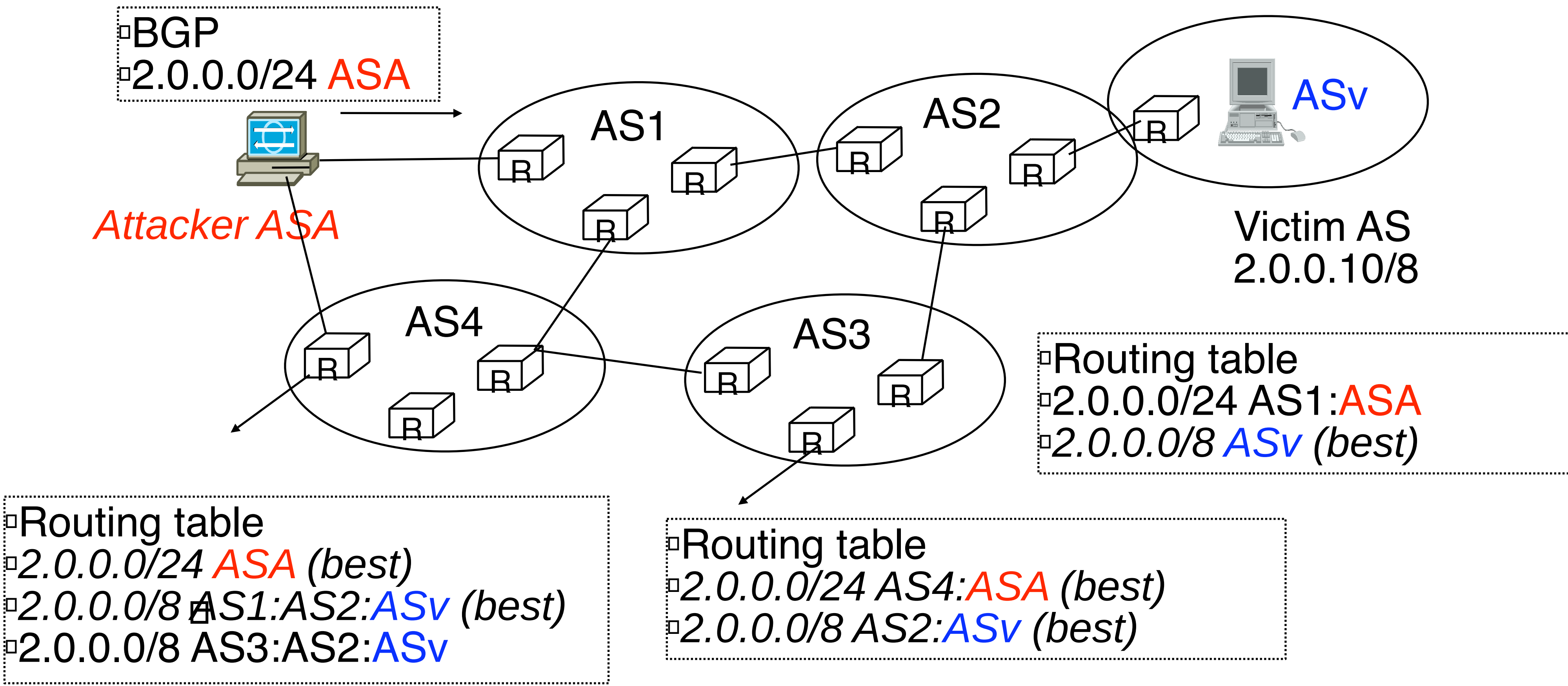
- First attack
  - Advertise the same prefix as the victim



- Result
  - Traffic from distant sources will be redirected to attacker

Invalid advertisements (2)

- Second attack
  - Advertise *a more specific* prefix than the victim



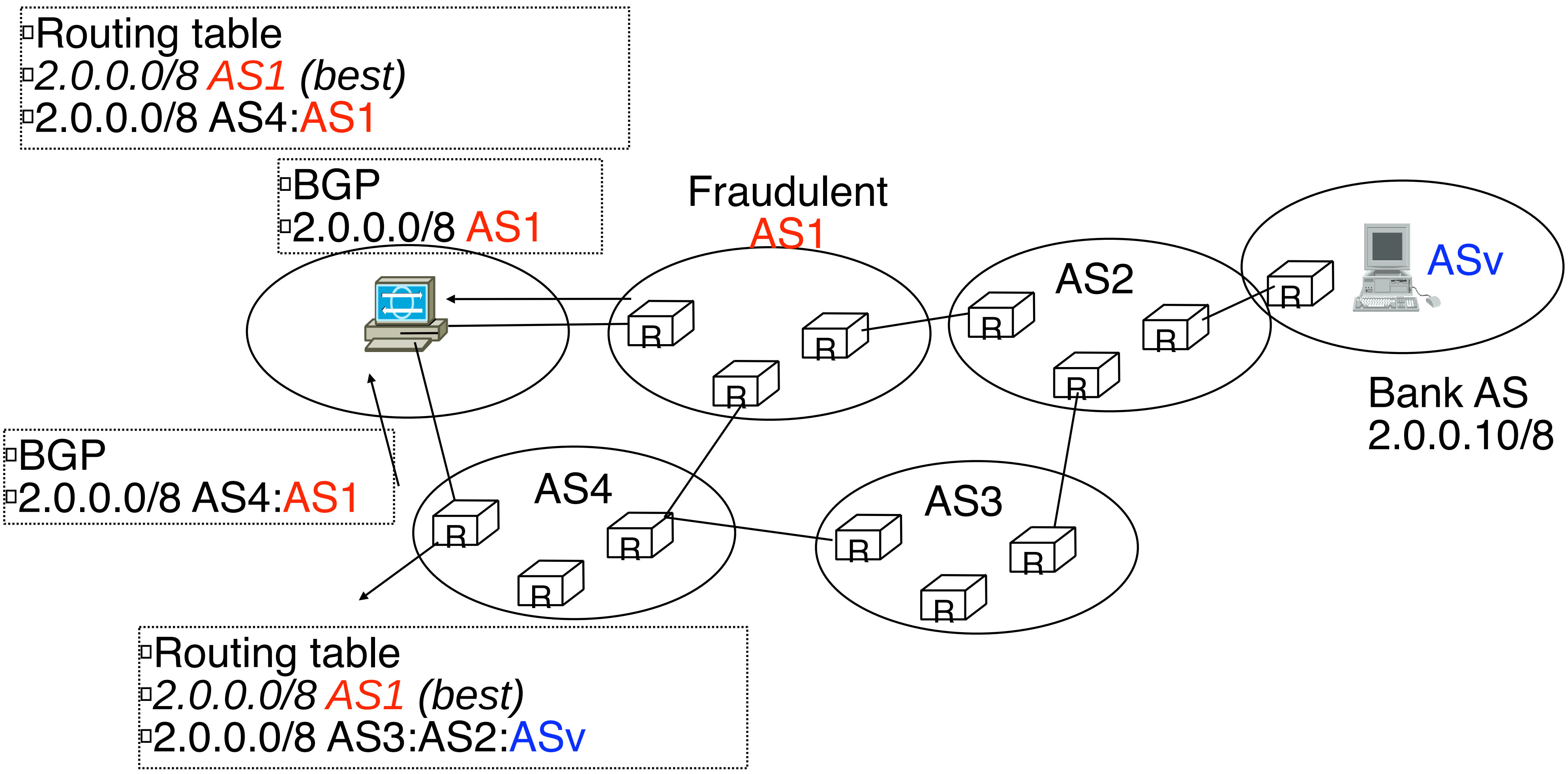
- Result
  - Traffic from *all* sources to specific prefix is redirected to the attacker

Security issues with interdomain routing (2)

- Problem
  - Any BGP router can change the content of a received BGP UPDATE
    - Add its own AS number in the AS-Path
    - Add/change BGP communities, local-pref, ...
  - Possible attack
    - Remove some ASes from received AS-Path
      - AS-Path is used to select the best route, thus received route has better chance of being selected
      - AS-Path is also used to detect routing loops, removing an AS number may cause interdomain loops
  - Possible consequence
    - By manipulating received UPDATE messages, a BGP router could attract packets for more destinations

How can an ISP attract more packets ?

- Possible attack
  - Fraudulent AS strips received AS-Path



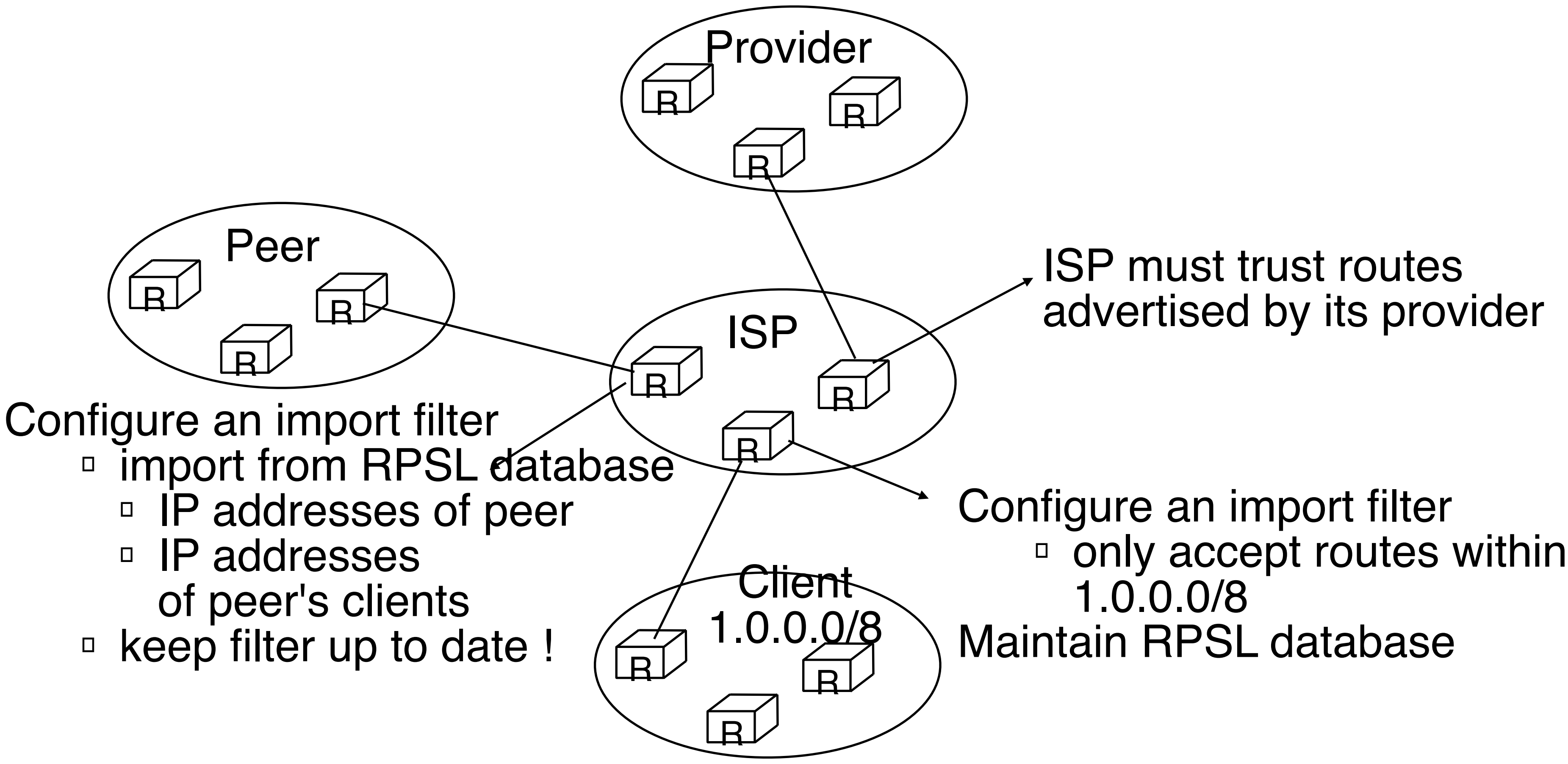
- This attack suffers from several problems :
- If the fraudulent AS strips all received AS Paths, then its peers and customers will easily notice the attack. However, there are today almost 200.000 BGP routes in the Internet and a Fraudulent AS could be interested in faking a small number of routes. This would be sufficient to collect all packets sent to banks ore large amounts of traffic in practice as a typical network will exchange lots of packets with only a small number of ASes
- By striping the AS-Path, the fraudulent AS blocks the loop detection mechanism used by BGP. This may cause interdomain loops and such loops could be more easily detected. This problem can be avoided by using the AS-Sets attribute supported by BGP. An AS-Set is an unordered list of AS numbers that are used to indicate the transit ASes for a given route under specific circumstances. This AS-Set is used to perform interdomain loop detection, but the BGP decision process will consider an AS-Set as having a length of one AS. With AS-Sets, AS1 would advertise {AS1,AS2,ASv} (a path with a length of one) while AS4 would advertise AS4:{AS1,AS2,ASv} (a path with a length of two).

Security issues with interdomain routing (3)

- Problem
  - A BGP session runs over a TCP connection
    - TCP connection between the two routers on port 179
  - A BGP session is considered as closed and all routes are withdrawn when TCP connection fails
- Security risks
  - If an attacker can inject valid BGP messages on an existing session, he could inject routes on the entire Internet
  - If an attacker could force a TCP connection to fail, he could cause large disruptions

Current solutions to improve security of interdomain routing

- Filter invalid routes
  - Whenever possible, routers should verify the validity of the routes received



Several RPSL databases exist. They are usually maintained by Regional Internet Registries such as RIPE in Europe. Some ISPs maintain their own RPSL database and force their customers to use this database.

Of course, the security of this verification depends on the security of the RPSL database...

Current solutions to improve security of interdomain routing (2)

- Monitoring
  - Collect the advertisements for important prefixes received by distant ASes
  - Verify that the origin AS is always correct
- Existing BGP monitors
  - Routeviews
  - RIPE RIS, myAS
- Limitations
  - Monitoring allows to detect problems, but solving them usually require cooperation between ISPs

Routeviews is a service maintained by the University of Oregon. It is composed of several BGP routers that receive the BGP routes advertised by a few tens of ISPs mainly located in the USA. Routeviews provides realtime access to the collected data as well as large archives :

http://www.routeviews.org

The Routing Information Service from RIPE is another service that collects BGP routes advertised by multiple ISPs, mainly inside Europe. RIPE has installed one route collector on many large Internet Exchange Points and collects all the BGP messages advertised by the ISPs attached to this IXP. RIPE also provides a service to ISPs where they can be informed immediately by SMS or email when one of their prefixes appears to be originated by another AS based on the BGP messages collected at the various collector.

http://www.ripe.net

Current solutions to improve security of interdomain routing (3)

- Protection of the TCP connections
  - MD5 option
    - each BGP peer is configured with a password
    - each TCP segment contains a keyed MD5 hash
  - iBGP sessions
    - configure iBGP sessions between loopback addresses inside same IP prefix
    - install packet filters on border routers to drop packets sent to/from this prefix
  - eBGP sessions
    - send TCP segments inside IP packets with TTL=255
    - only accept TCP segments received from valid IP address and with TTL=255

The TCP-MD5 option used to protect BGP sessions is described in :

A. Heffernan, Protection of BGP Sessions via the TCP MD5 Signature Option. . August 1998. RFC2385

The TTL Security Hack is described in :

V. Gill, J. Heasley, D. Meyer. , The Generalized TTL Security Mechanism (GTSM). February 2004. RFC3682

BGP security extensions

- Several extensions are being developed to secure BGP, but they are not deployed
  - S-BGP
    - assumes that two PKIs will be deployed to
      - First PKI is used to certify allocation of IP addresses
      - Second PKI is used to certify that AS numbers belong to organisations and also for routers
        - allows ASes to sign the AS-Path that they announce
    - main concern is the CPU cost
  - SoBGP
    - A simpler and more pragmatic approach
      - A PKI and certificates are used to prove prefix ownership
      - A database of inter-AS relations is built and used to validate the received AS-Paths
- SIDR working group is developing certificates to prove owernship of addresses

S-BGP is described in several papers, including :

S. Kent, C. Lynn, K. Seo, Secure Border Gateway Protocol (S-BGP), IEEE Journal on Selected Areas in Communications, Vol 18, N4, 2000

SoBGP is being developed within IETF. A tutorial description of SoBGP may be found in :

R. White, Securing BGP through secure origin BGP,Internet Protocol Journal Sept. 2003