Network Address Translation (NAT)

- References:
  - “Peer-to-Peer (P2P) communication across middleboxes”, Internet Draft, http://midcom-p2p.sourceforge.net/draft-ford-midcom-p2p-01.txt.
  - [NAT-PT] G. Tsirtsis and P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)", RFC 2766, February 2000.
  - [NAT-TRAD] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.
  - [STUN] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.
- Acknowledgement:
  - Some pictures in this file are made by Ethan Lin (林君鴻)

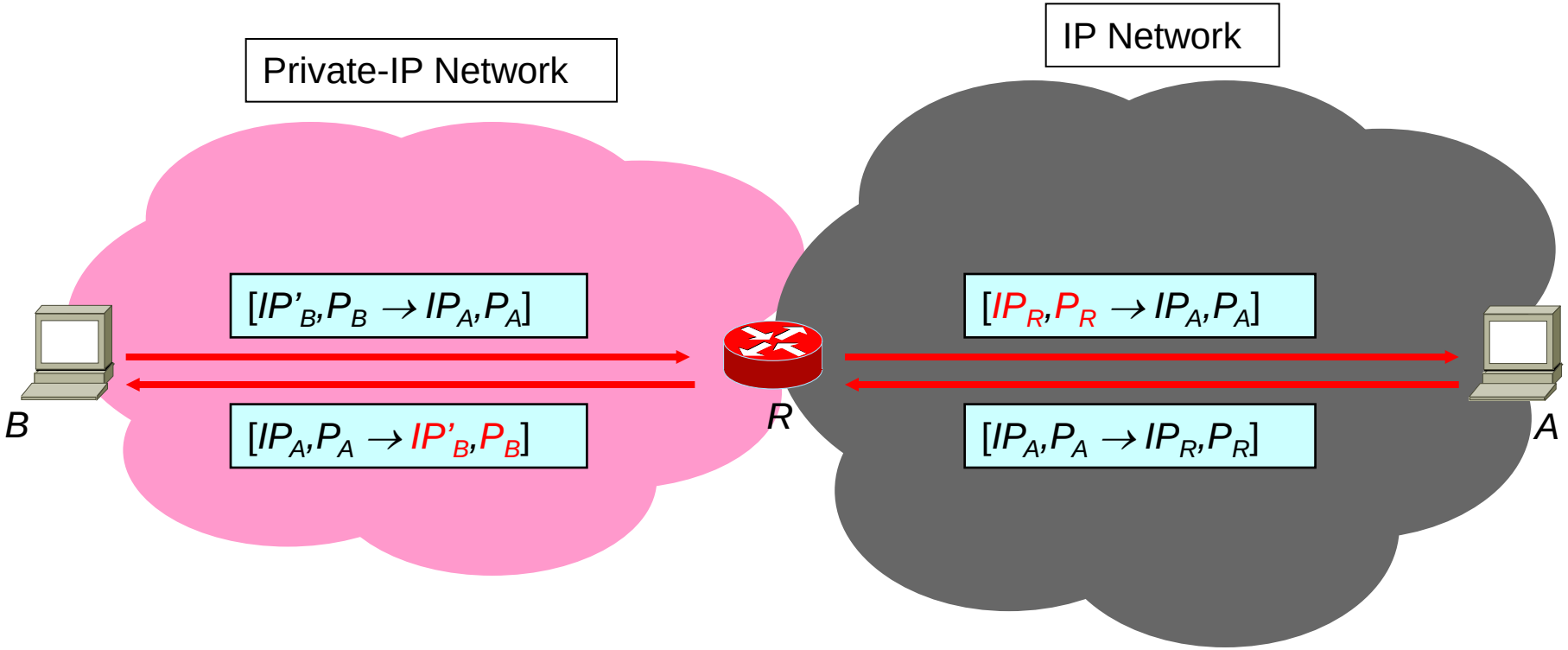
Firewalls vs. NAT

- Common:
  - A firewall restricts communication between a private internal network and the public Internet, typically by dropping packets that are deemed unauthorized.
- Difference:
  - Firewalls
    - A firewall does not modify the IP address and TCP/UDP port information in packets crossing the boundary.
  - NAT:
    - A NAT modifies the header information in packets flowing across the boundary.

RFC1918 Private Network Addresses

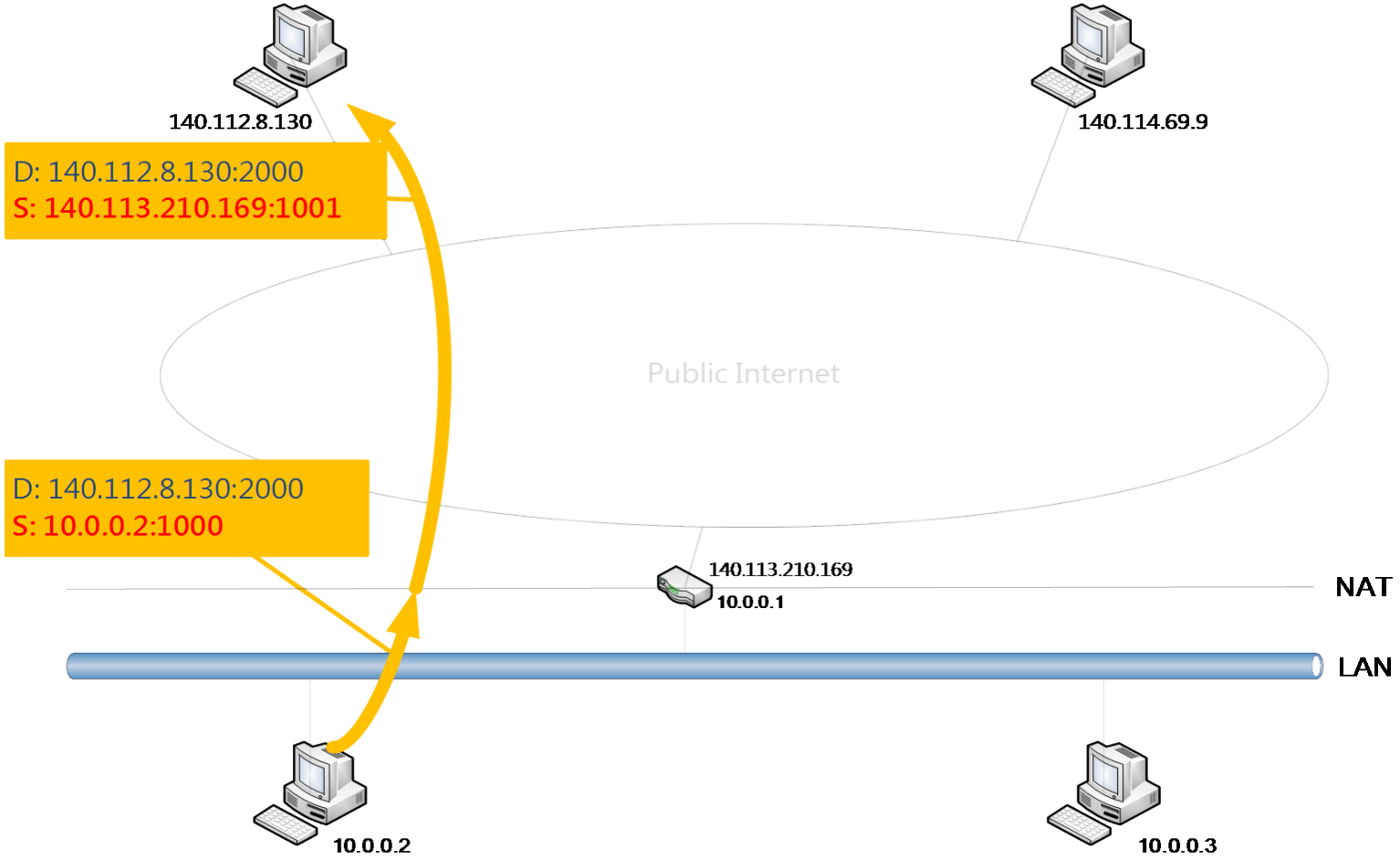
- 256 Class C Networks:
  - 192.168.0.0 thru 192.168.255.0
- 16 Class B Networks:
  - 172.16.0.0 thru 172.31.0.0
- 1 Class A Network:
  - 10.0.0.0

Address Translation



$[IP'_B,P_B \rightarrow IP_A,P_A]$	$[IP_R,P_R \rightarrow IP_A,P_A]$
$[IP_A,P_A \rightarrow IP'_B,P_B]$	$[IP_A,P_A \rightarrow IP_R,P_R]$

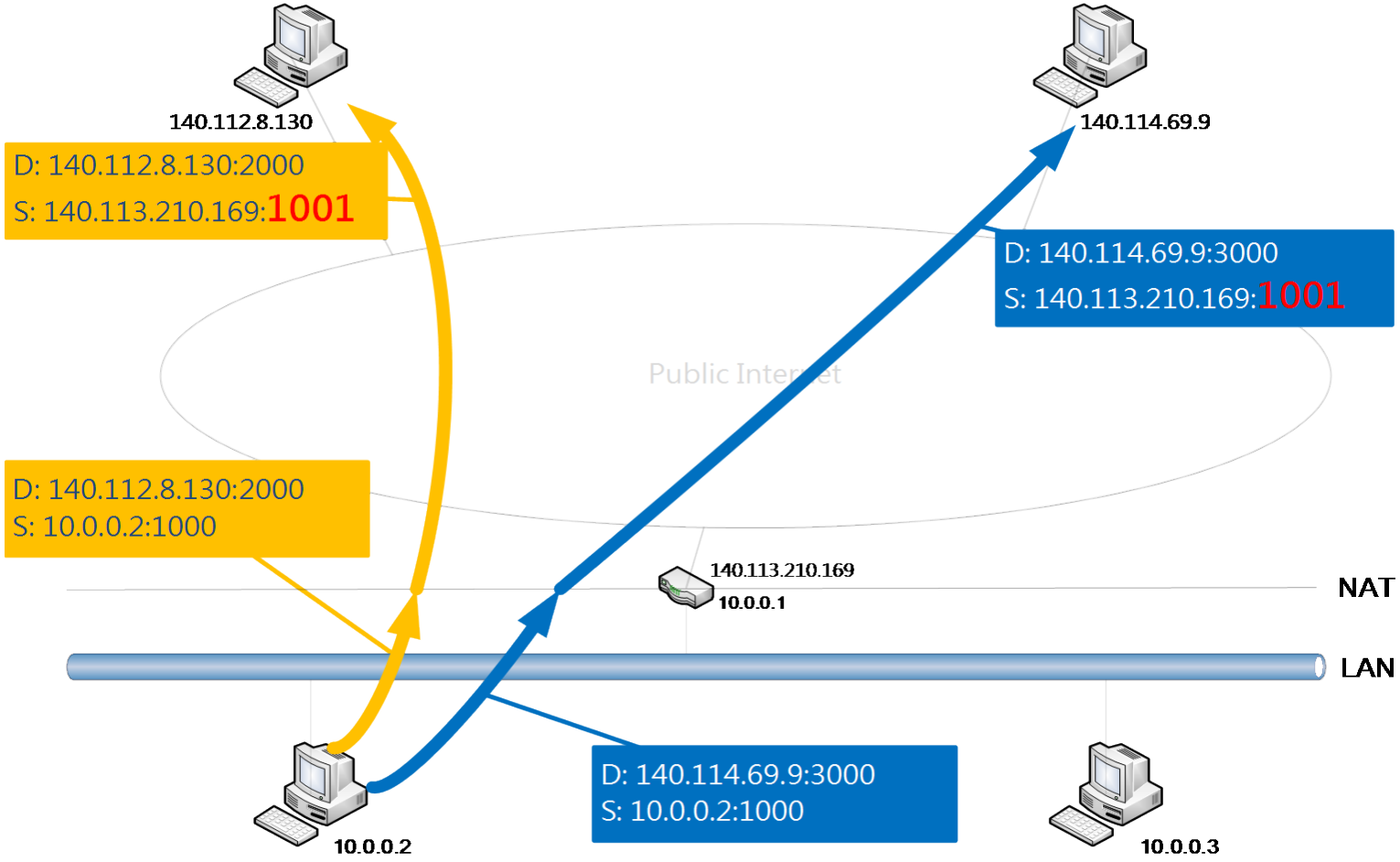
How NAT Works?



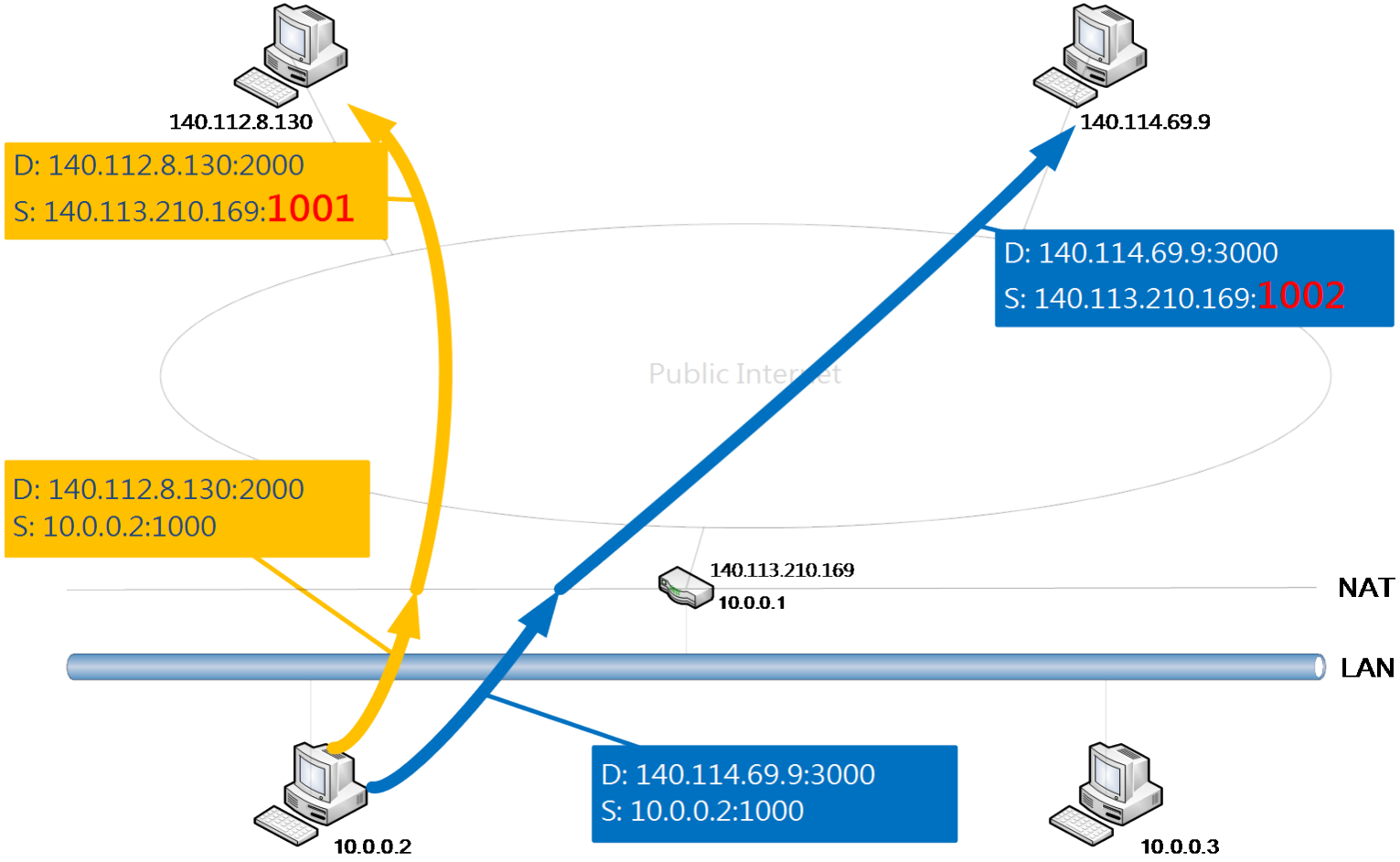
NAT Type

	Incoming scheme 1	Incoming scheme 2	Incoming scheme 3
Outgoing scheme A	Full Cone NAT	Restricted Cone NAT	Port Restricted Cone NAT
Outgoing scheme B	Symmetric NAT		

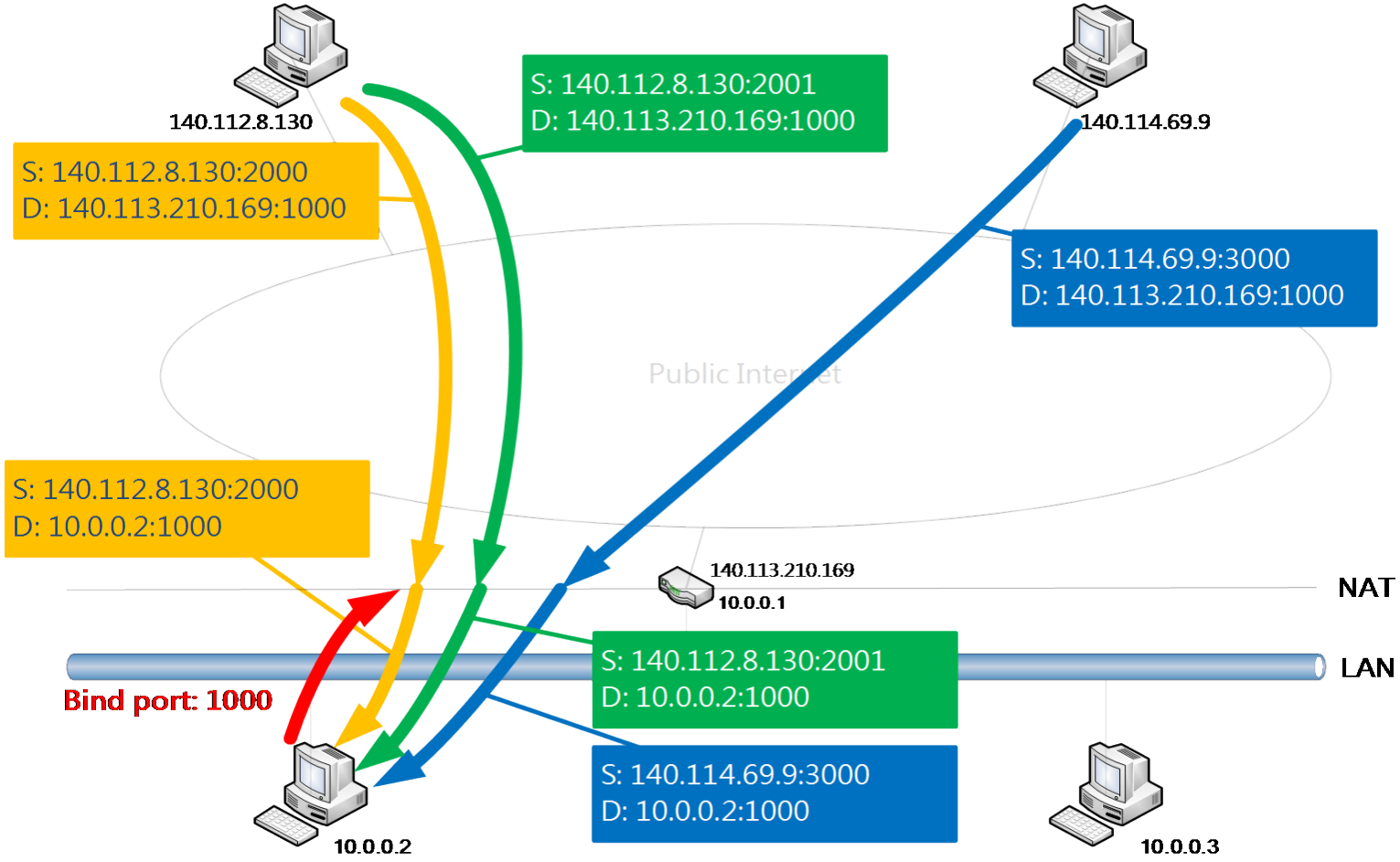
NAT Outgoing Scheme A



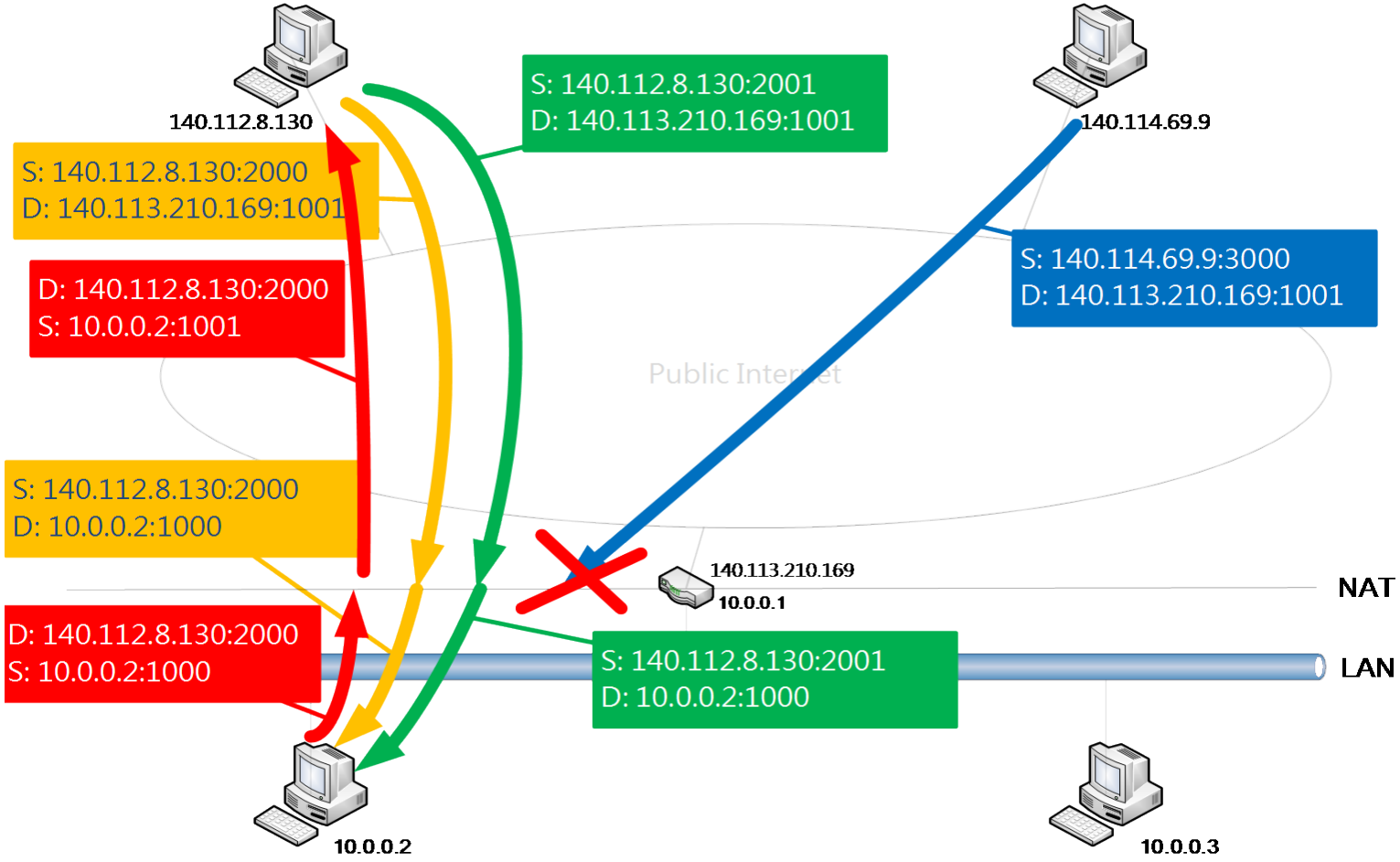
NAT Outgoing Scheme B



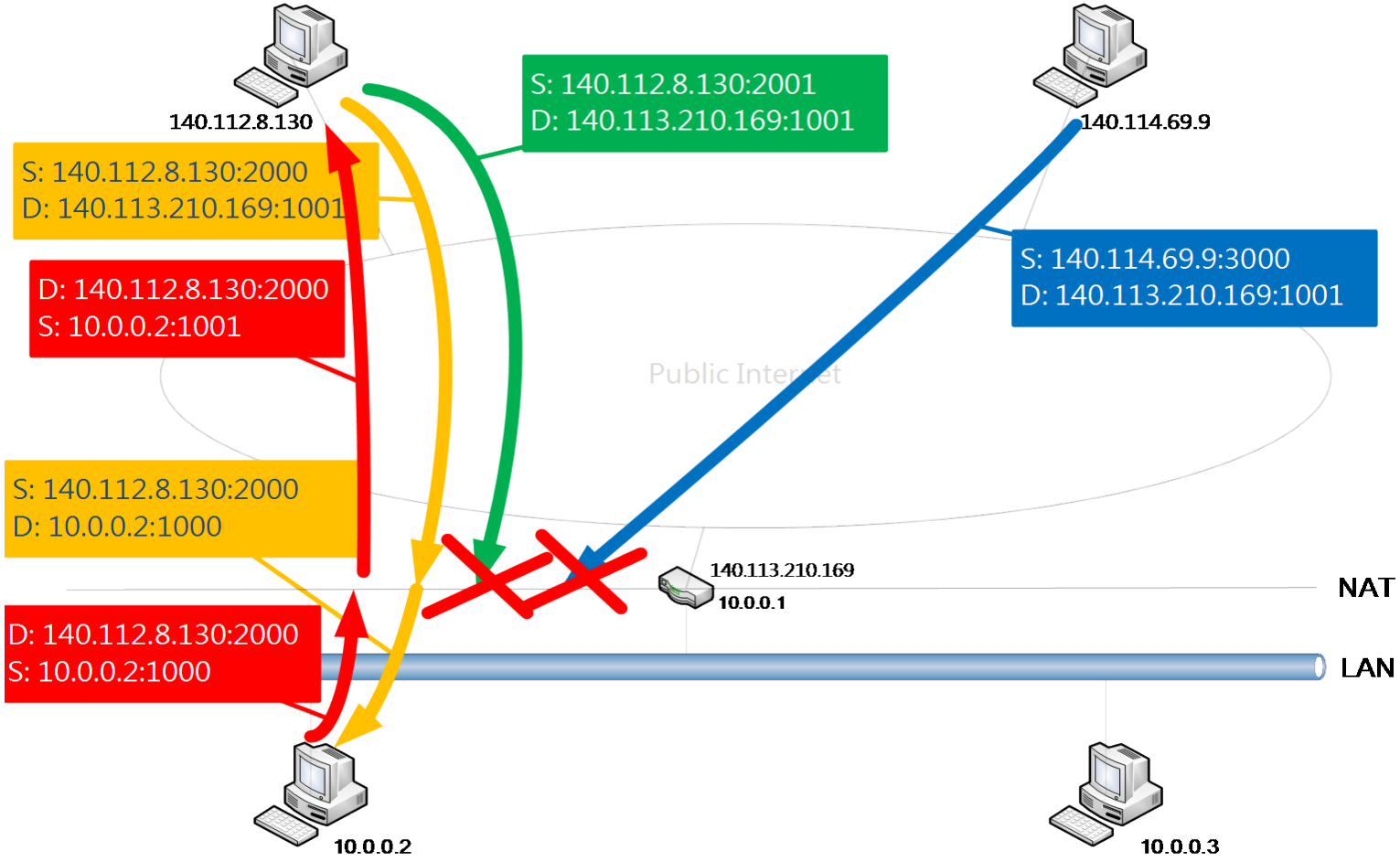
NAT Incoming Scheme 1



NAT Incoming Scheme 2



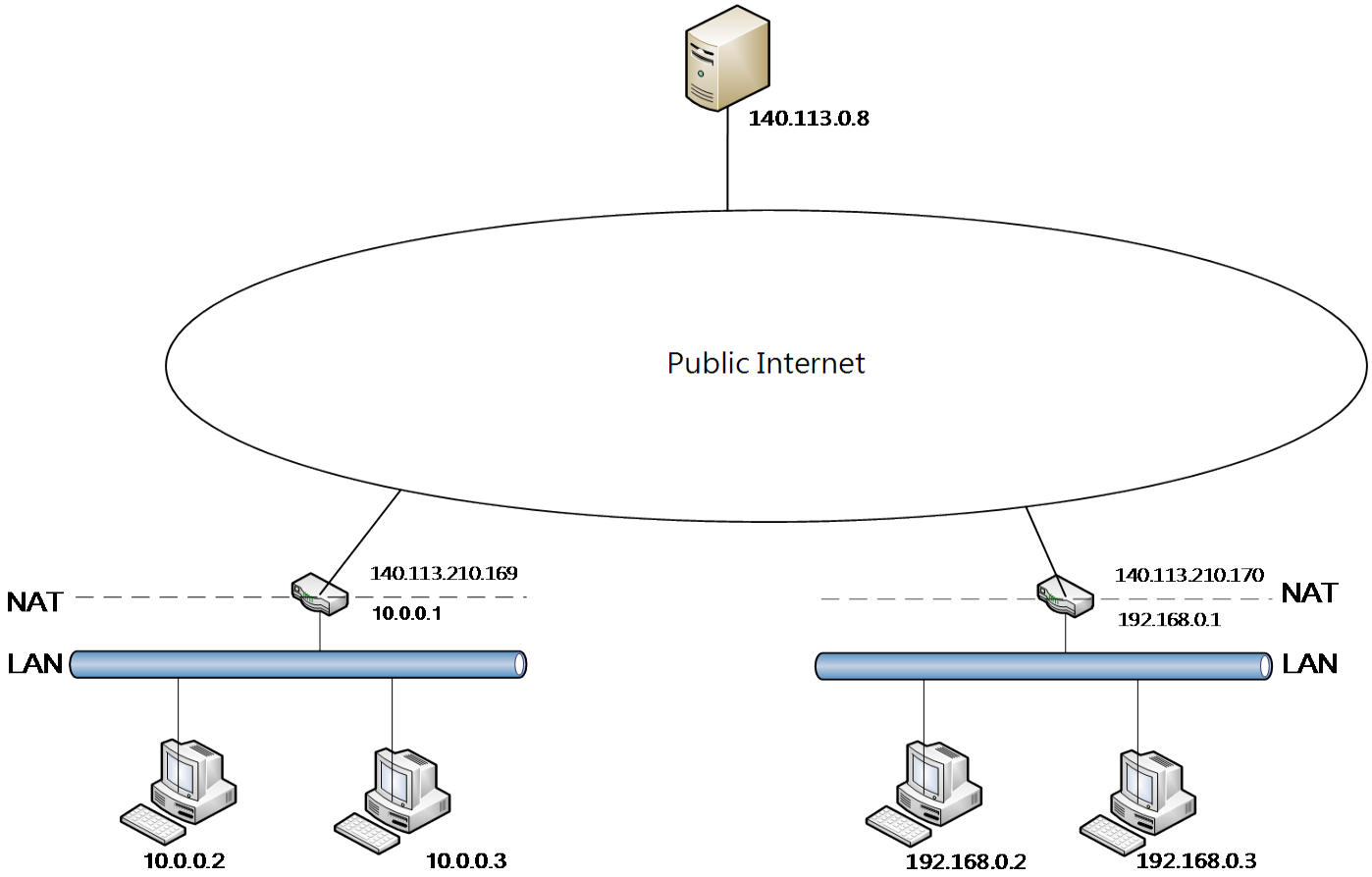
NAT Incoming Scheme 3



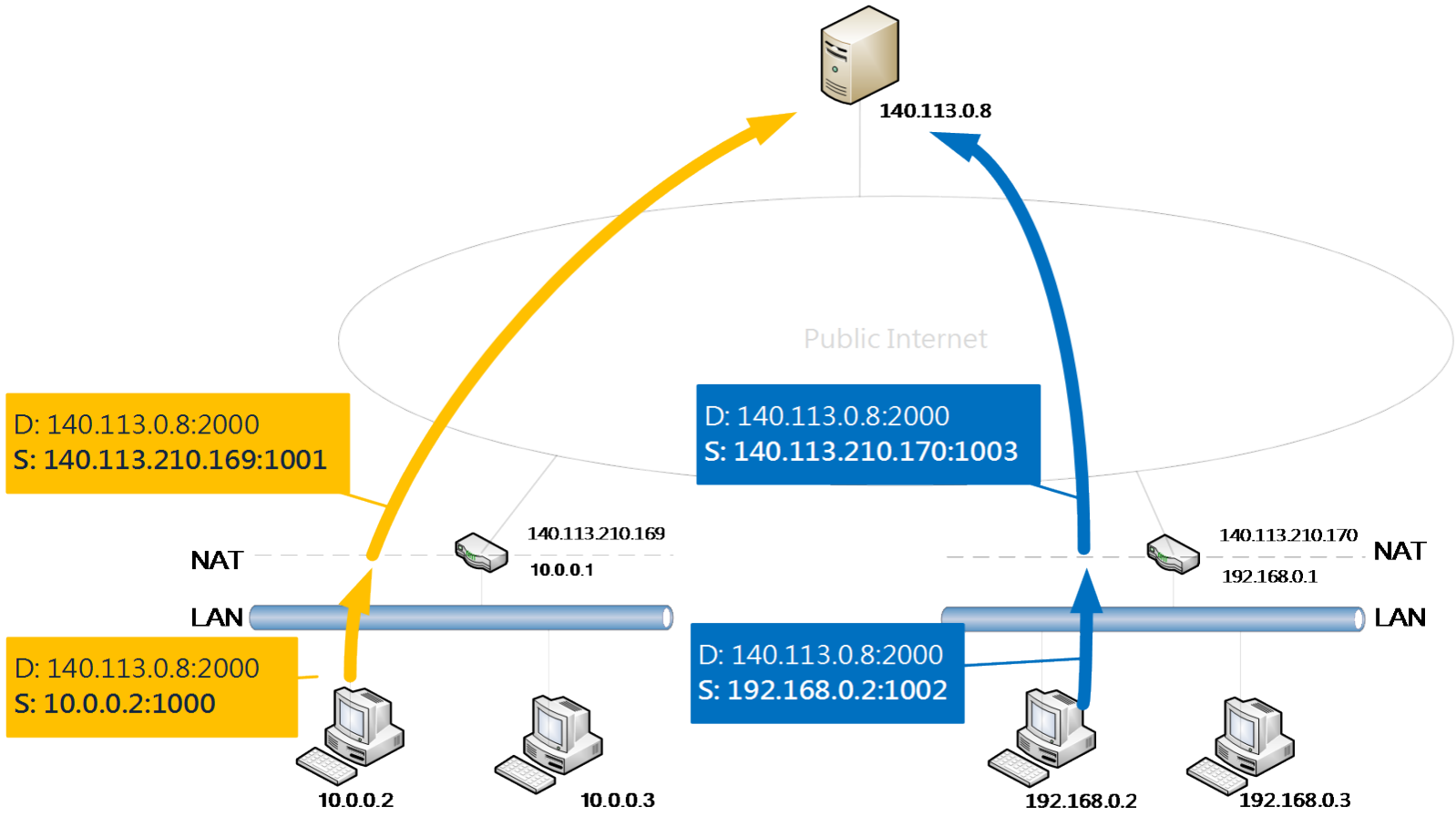
How to Penetrate?

- Depend on the types of NAT systems.
  - Full Cone NAT
    - Direct connect without any problem.
  - Restricted NAT
    - Need to use UDP hole punching
  - Port Restricted NAT
    - Need to use UDP hole punching
  - Symmetric NAT
    - Need to use SuperNode

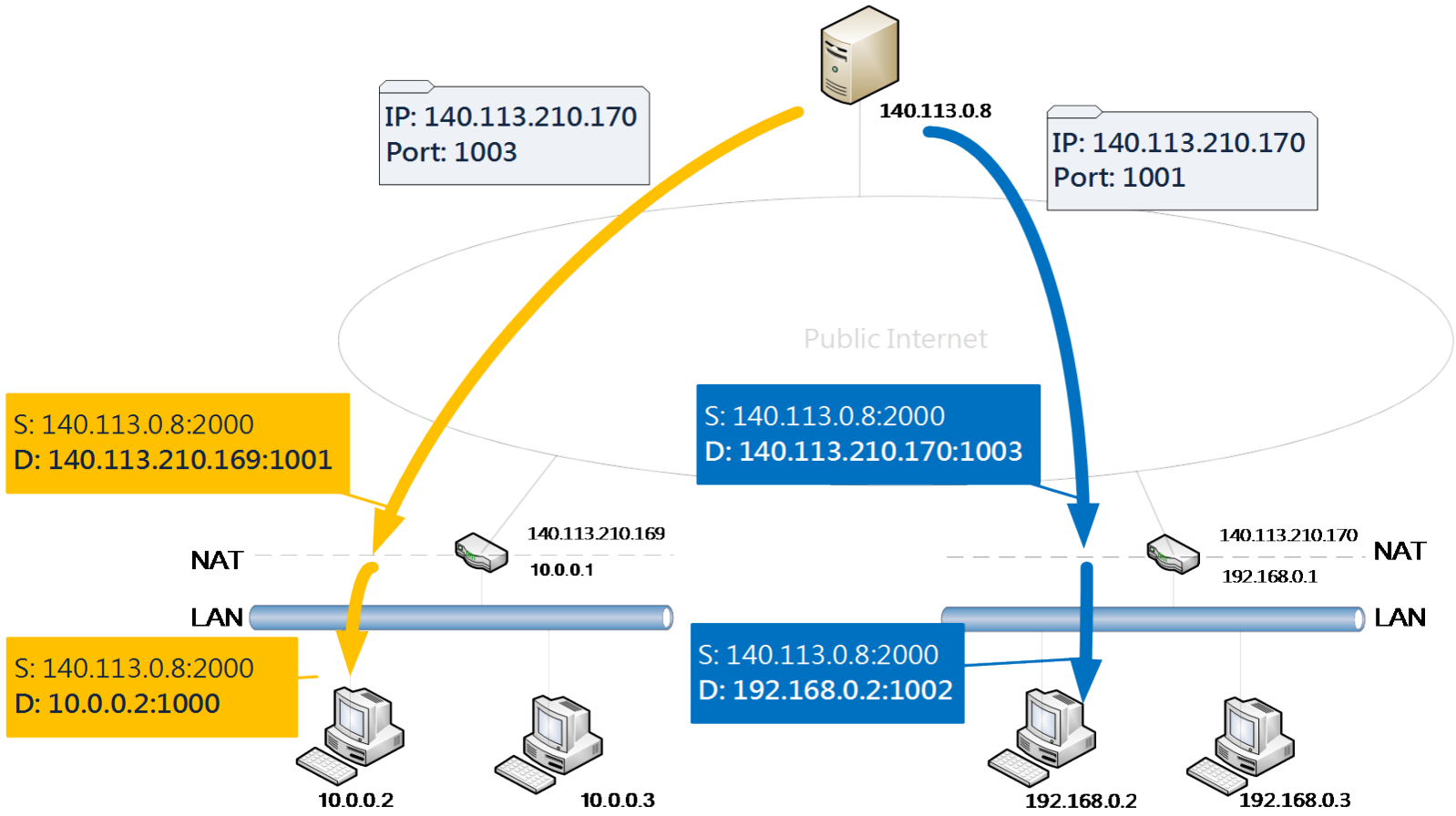
UDP Hole Punching



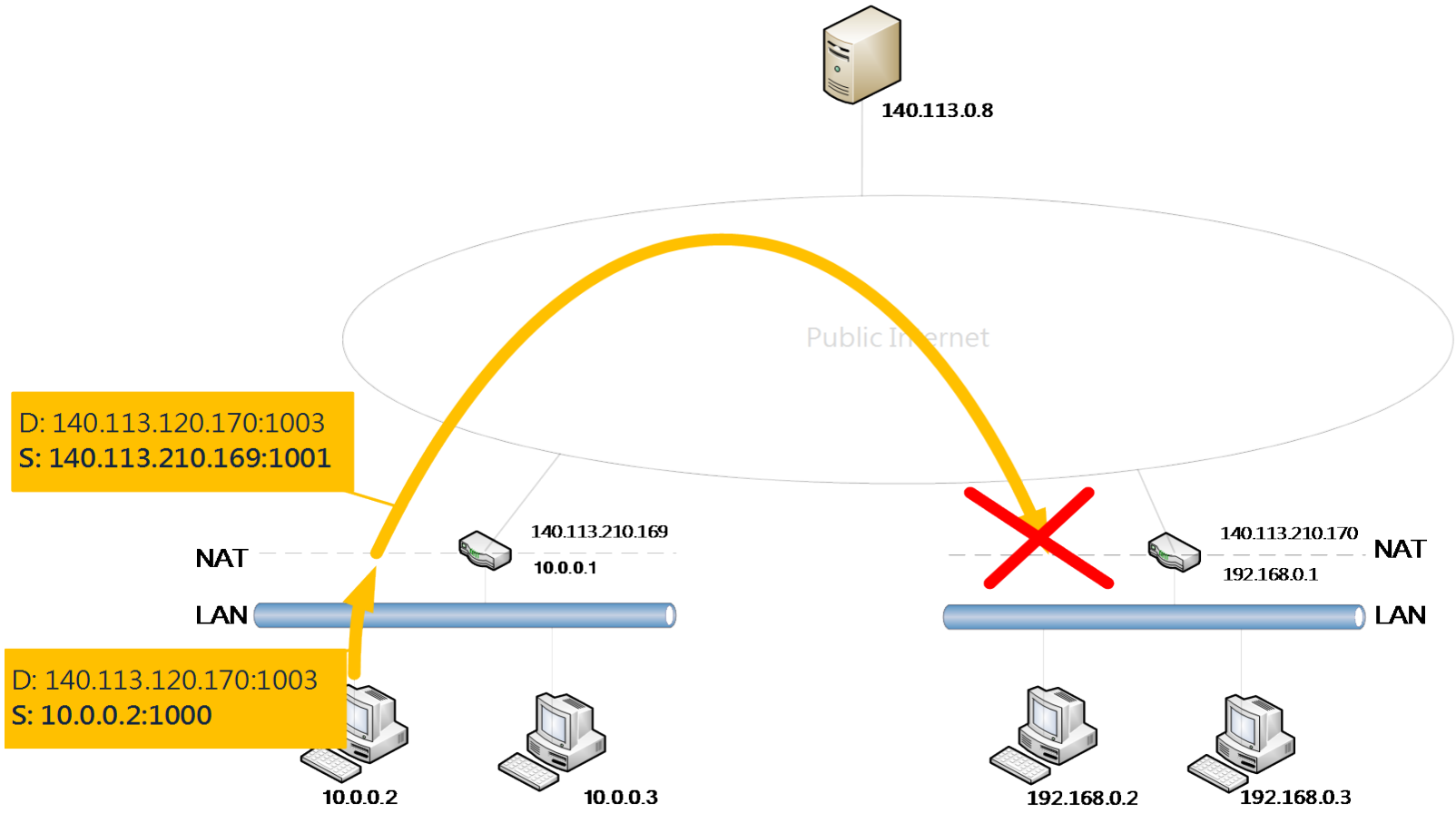
UDP hole punching



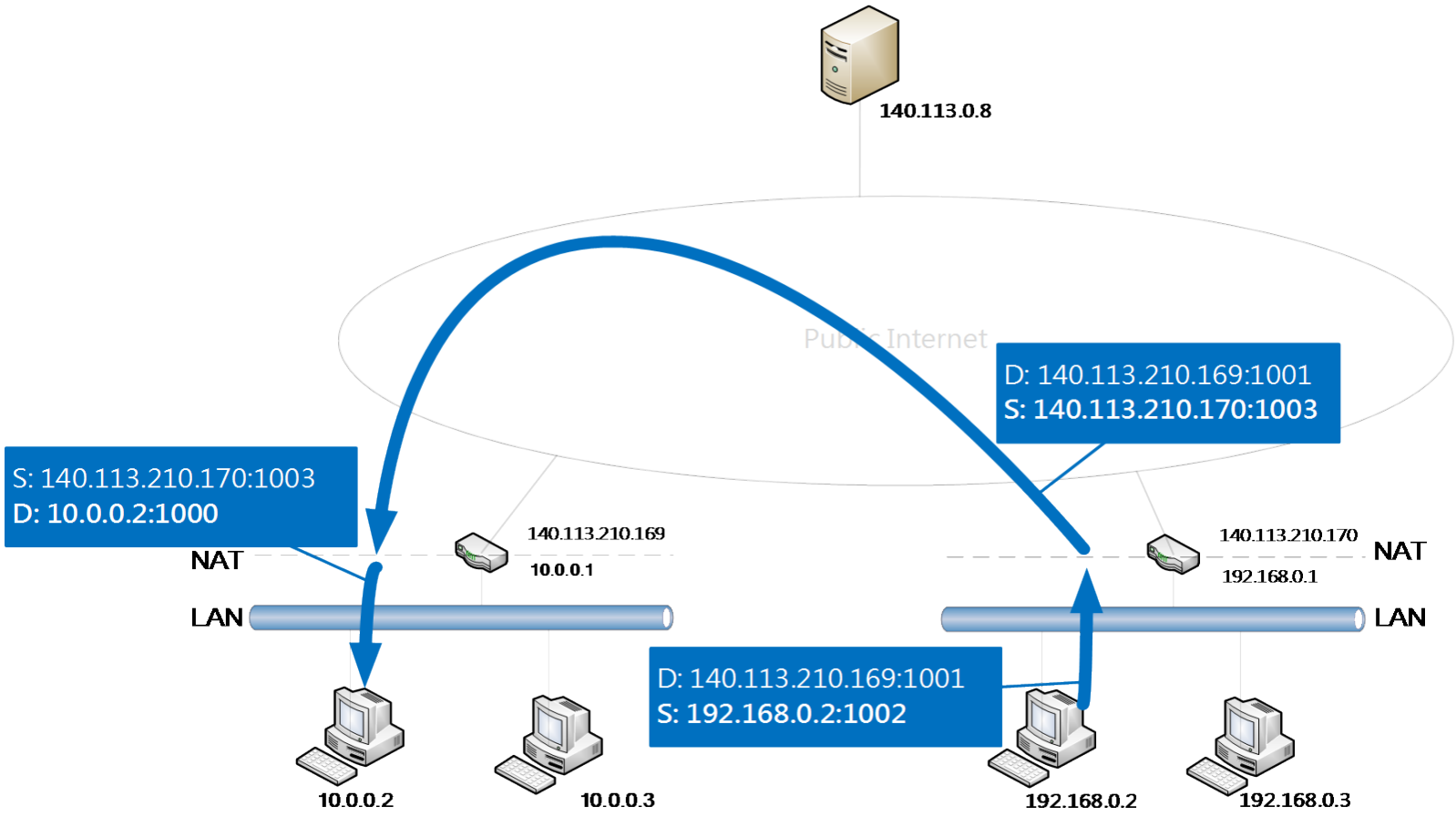
UDP hole punching



UDP hole punching



UDP hole punching



STUN protocol

- discover the presence and types of NATs and firewalls between them and the public Internet

WebSocket

- References:
  - RFC 6455
  - The WebSocket API - W3C draft specification of the API
  - The WebSocket protocol - Internet-Draft published by the IETF HyBi Working Group

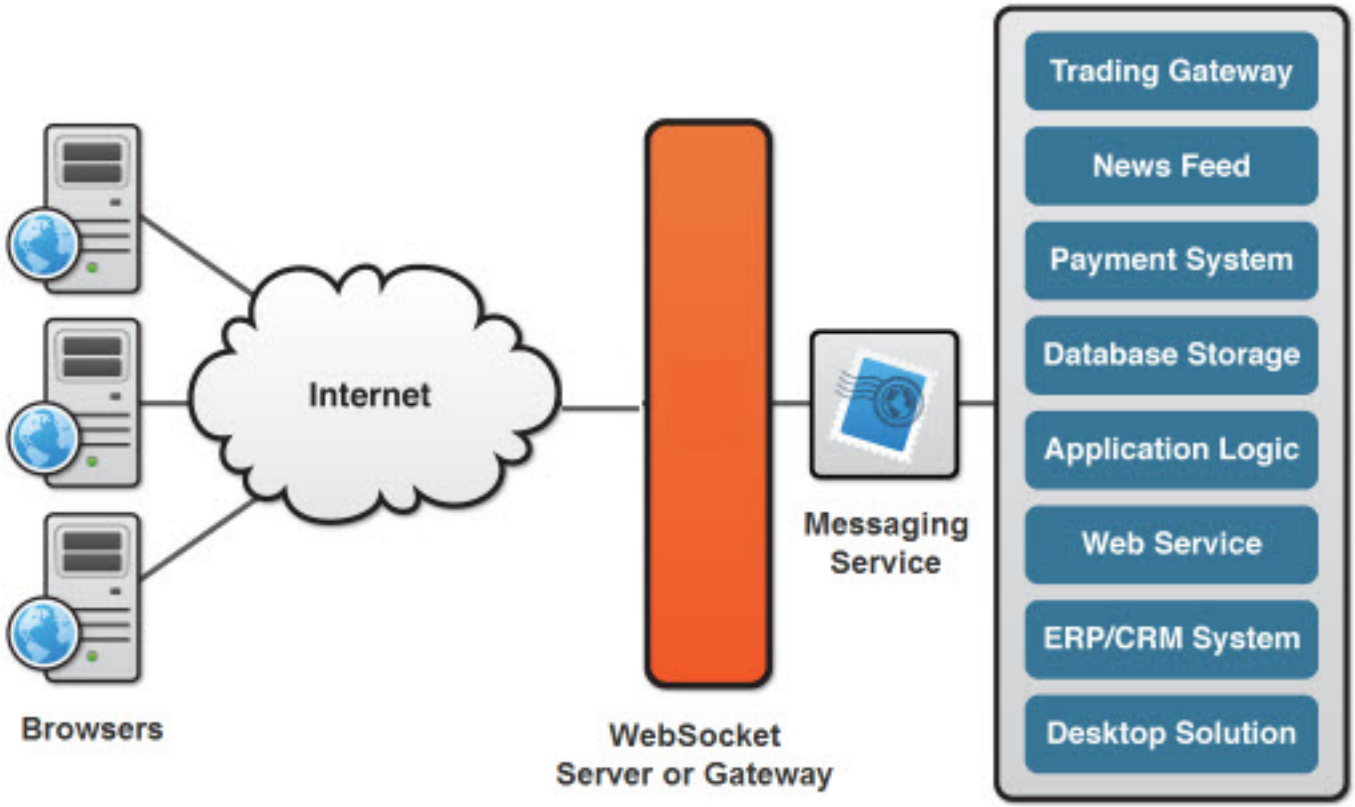
WebSocket

- A technology providing for bi-directional, full-duplex communications channels, over a single TCP socket.
- Problem:
  - Ordinary TCP connections to port numbers other than 80 are frequently blocked by administrators outside of home environments.
- Purpose:
  - A way to overcome these restrictions and provide similar functionality with some additional protocol overhead while multiplexing several WebSocket services over a single TCP port.

WebSocket

- Standardization:
  - The WebSocket API
    - Standardized by the W3C,
  - WebSocket protocol
    - Standardized by the IETF as RFC 6455.
- Clients:
  - Browsers: Firefox 6, Google Chrome 14, Opera 11, and Safari 5,
  - Mobile: mobile version of Safari in iOS 4.2, the BlackBerry Browser in OS7 supports WebSocket.

Architecture



WebSocket Protocol

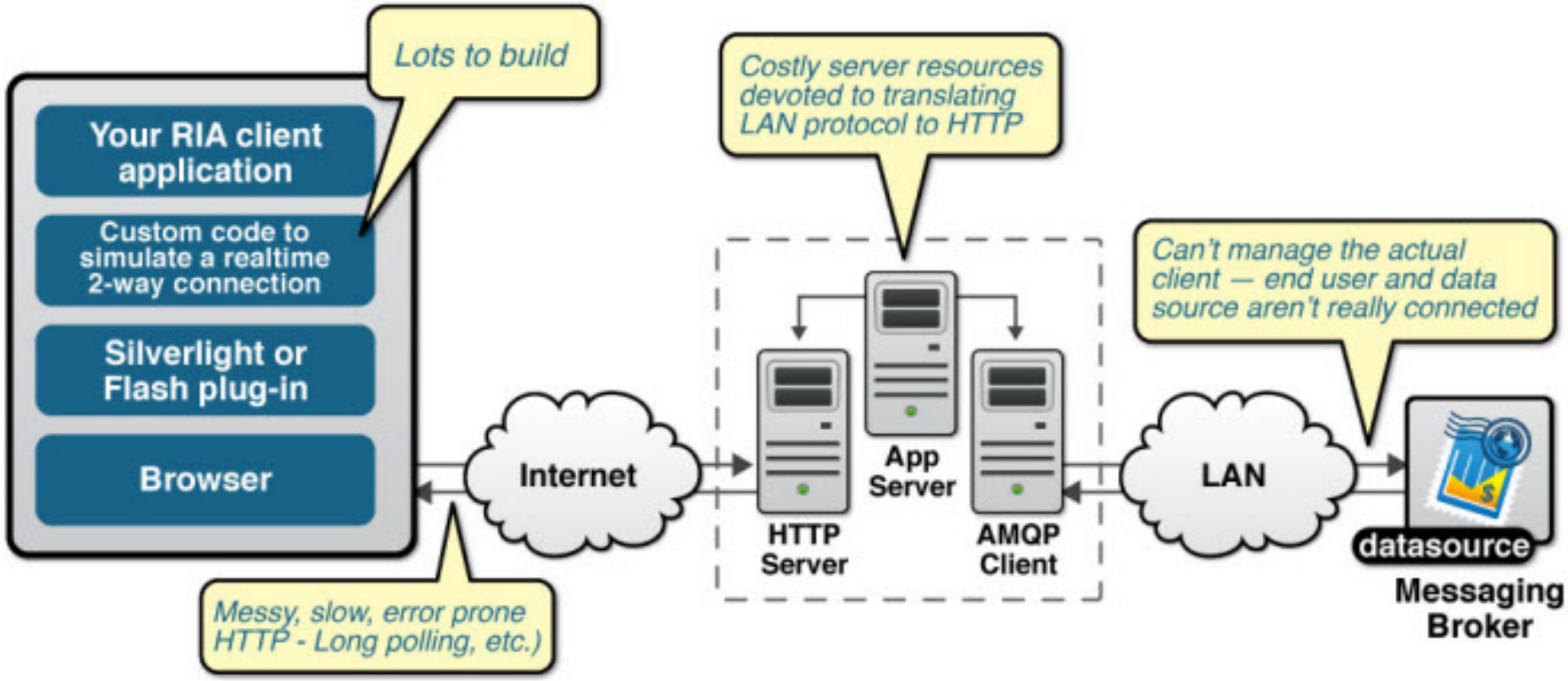
- Browser:

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

- Server:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

An Application



Using the HTML5 WebSocket API

- Create a web socket.

```
var myWebSocket = new WebSocket("ws://www.websockets.org");
```

- Set up event handlers for receiving messages.

```
myWebSocket.onopen = function(evt) { alert("Connection open ..."); };
myWebSocket.onmessage = function(evt) { alert( "Received Message: " + evt.data); };
myWebSocket.onclose = function(evt) { alert("Connection closed."); };
```

- Send messages.

```
myWebSocket.send("Hello Web Sockets!");
myWebSocket.close();
```