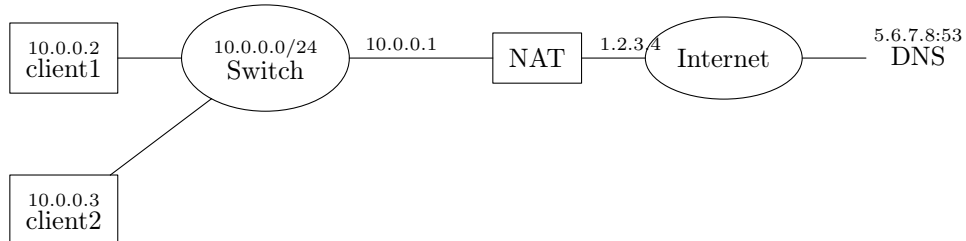


# (Network Address Translator)NAT

NAT זה מנגנון שמטרתו לחסוך בכתובות IP. NAT מאפשר לפצל כתובות IP, כך שבתוך הרשת לכל מחשב תהיה כתובת משלו, ומבחוץ יתיחסו לכולן כאל כתובת אחת.



כש client1 רוצה לשלוח חבילה ל-DNS, הוא שולח

SrcIP	: 10.0.0.2
SrcPort	: 2222
DestIP	: 5.6.7.8
DestPort	: 53

ה NAT פותח רשומת המרה -

10.0.0.2	:	2222	
SrcIP	SrcPort	DestIP	DestPort
1.2.3.4	3333	5.6.7.8	53

- ושולח את החבילה המקורית בתור

SrcIP	: 1.2.3.4
SrcPort	: 3333
DestIP	: 5.6.7.8
DestPort	: 53

לכן שרת ה-DNS מחזיר תשובה עם

SrcIP	: 5.6.7.8
SrcPort	: 53
DestIP	: 10.0.0.2
DestPort	: 2222

וה NAT ממיר את זה ל

SrcIP	: 5.6.7.8
SrcPort	: 53
DestIP	: 1.2.3.4
DestPort	: 3333

מגיע בחזרה ל client1.

כל חבילה שמגיעה ואין לה רשומה ב-NAT, היא נזרקת.

## יש כמה סוגים של NATים:

### Full Cone NAT

עובד ע"פ DestIP:DestPort בלבד.

SrcIP	: 5.6.7.9
SrcPort	: 2456
DestIP	: 1.2.3.4
DestPort	: 3333

אם יש מחשב אחר באינטרנט ששולח חבילה לאותו מקום:

- אז למרות שזה משולח אחר, זה עדיין יגיע ל client1, בגלל שיש את ה-DestIP ואת ה-DestPort שרשומים ב-NAT.

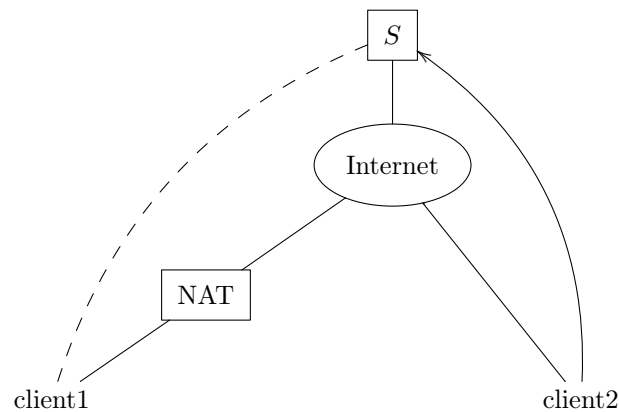
### Port Restricted NAT

מסתכל גם לפי SrcIP:SrcPort.

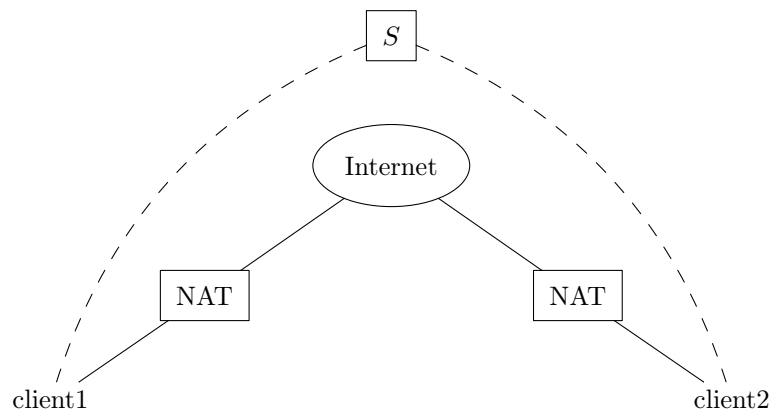
אם מחשב אחר שולח חבילה לאותו מקום, אז זה לא יגיע לclient1 - כי זה מגיע מSrcIP:DestIP אחר.

## Peer to Peer

בד"כ לשרתים יש IP משלהם, ולקוחות יוזמים פניות לשרתים(ולא להפך), ולכן אין בעיה - אבל כן יכולה להיות בעיה באפליקציות Peer2Peer, כי אז לקוח פונה ללקוח אחר - אבל ללקוח שפונים אליו אין כתובת IP משלו. בשביל זה משתמשים בשרת:



ברגע שהלקוח הראשון מתחבר לשרת, השרת אומר לו שהלקוח השני רוצה לדבר איתו, ואז הלקוח הראשון מתחבר ישירות ללקוח השני. התהליך הזה נקרא *Hole Punching*. אבל מה קורה אם שניהם מאחורי NAT?



במקרה הזה, שני הלקוחות צריכים לשלוח חבילות אחד לשני. החבילות אמנם נזרקות, אבל נוצרים מיפויים בNATים, וכשהם ישלחו בחזרה הNATים ידע לנטב אותן. עדיין צריך שרת ששני הלקוחות יתחברו אליו בשביל שהוא יודיע להם מה הIP והPort אחד של השני.

# Reliable Data Transfer - RDT

UDP זה בעצם IP עם תמיכה בPortים - הוא לא נותן לנו שום הבטחה לסדר או תקינות של חבילות. אם רוצים את זה צריך להשתמש בTCP, אבל הוא יותר כבד. נסתכל על מנגנון יותר פשוט - RDT. נסתכל על הדורות השונים של RDT, ונראה מה כל דור מוסיף:

## • RDT 1.0

**הנחה:** ערוץ אמין  
**שולח:** שולח מידע  
**מקבל:** מקבל מידע

## • RDT 2.0

**הנחה:** יש שיבושים בdata  
**שולח:** checksum, Retransmission  
**מקבל:** ACK/NAK

## • RDT 2.1

**הנחה:** שיבושים בdata ובאישורים  
**שולח:** הוספת מספר סידורי  
**מקבל:** checksum

## הצעת שינוי

אם המספר הסידורי בחבילה לא מתאים, המקבל לא מקבל ושולח NAK. מה יקרה? נניח שאת החבילה הראשונה  $(m_0, 0)$  ויש שיבוש - אז שולחים NAK. אבל נניח שהNAK משתבש והופך לACK. אז שולחים את החבילה השנייה -  $(m_1, 1)$ . אבל המחשב המקבל עדיין מצפה ל $(*, 0)$ , ושולח NAK, ואז השולח שולח שוב את  $(m_1, 1)$  ונתקעים בלולאה אינסופית.