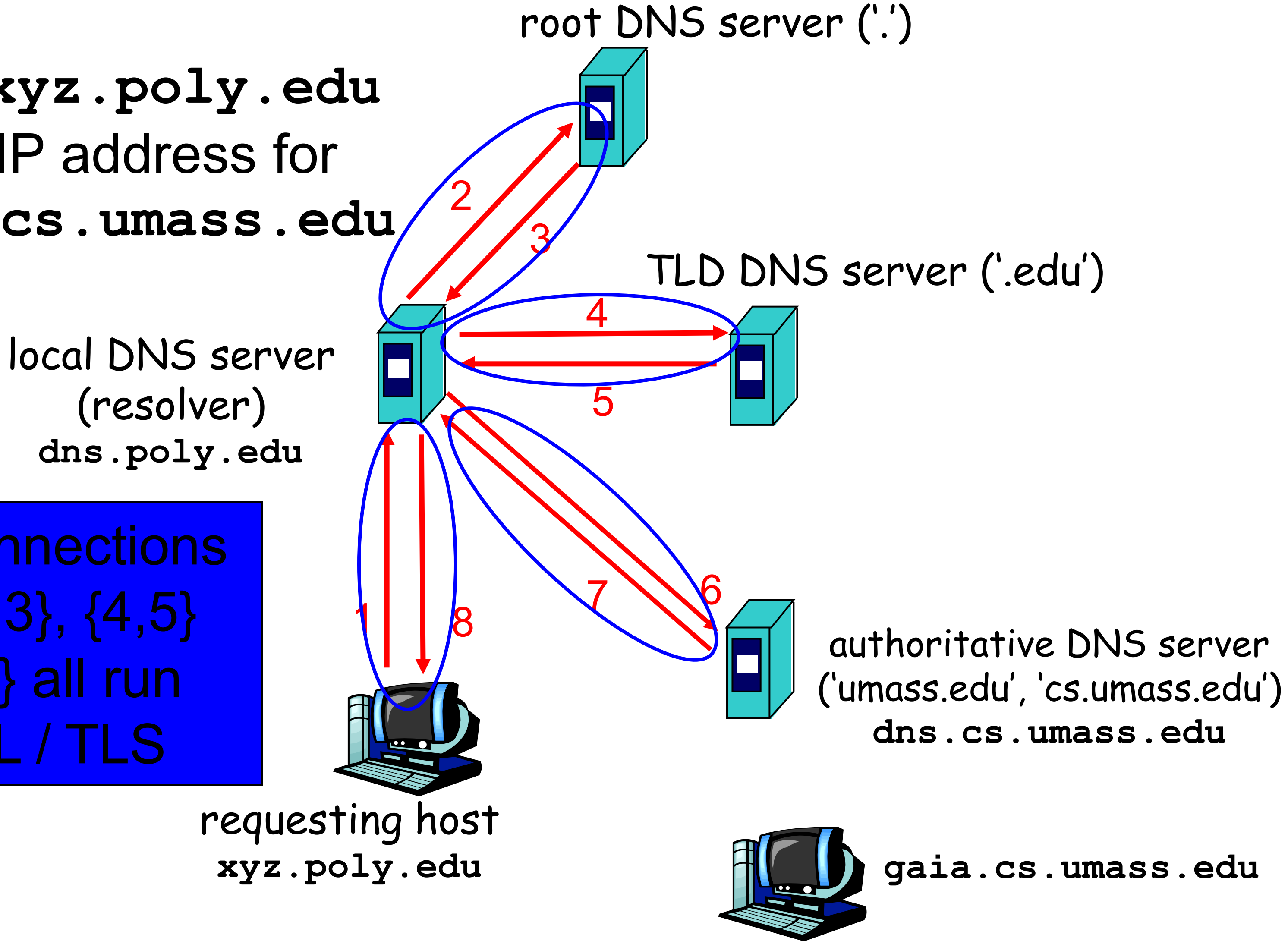
Securing DNS Lookups

- How can we ensure that when clients look up names with DNS, they can trust the answers they receive?
- Idea #1: do DNS lookups over TLS
  - (assuming either we run DNS over TCP, or we use "Datagram TLS")

Securing DNS using SSL / TLS

Host at **`xyz.poly.edu`** wants IP address for **`gaia.cs.umass.edu`**



Idea: connections {1,8}, {2,3}, {4,5} and {6,7} all run over SSL / TLS

Securing DNS Lookups

- How can we ensure that when clients look up names with DNS, they can trust the answers they receive?
- Idea #1: do DNS lookups over TLS
  - (assuming either we run DNS over TCP, or we use “Datagram TLS”)
  - Issues?
    - Performance: DNS is very lightweight. TLS is not.
    - Caching: crucial for DNS scaling. But then how do we keep authentication assurances?
- Idea #2: make DNS results like certs
  - I.e., a signed assertion, providing self-contained evidence who generated it (via a digital signature)

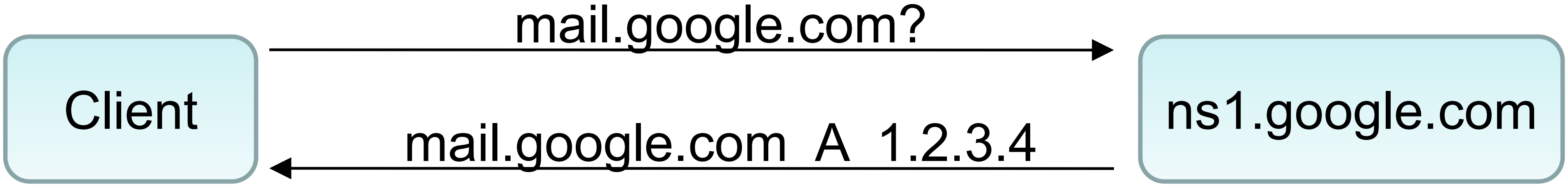
Operation of DNSSEC

- DNSSEC = standardized DNS security extensions currently being deployed

1. Suppose we look up `mail.google.com`
   - We get an answer from `google.com` nameserver (NS)
   - Plus: signature for answer (in Additional section) purportedly signed by `google.com` NS
2. Look up public key for `google.com` NS
   - That answer is signed by `.com` NS
3. Look up public key for `.com` NS
   - That answer is signed by root (‘`.`’) NS
4. Root NS’s public key is wired into our resolver

- All of these keys are cacheable

DNS:



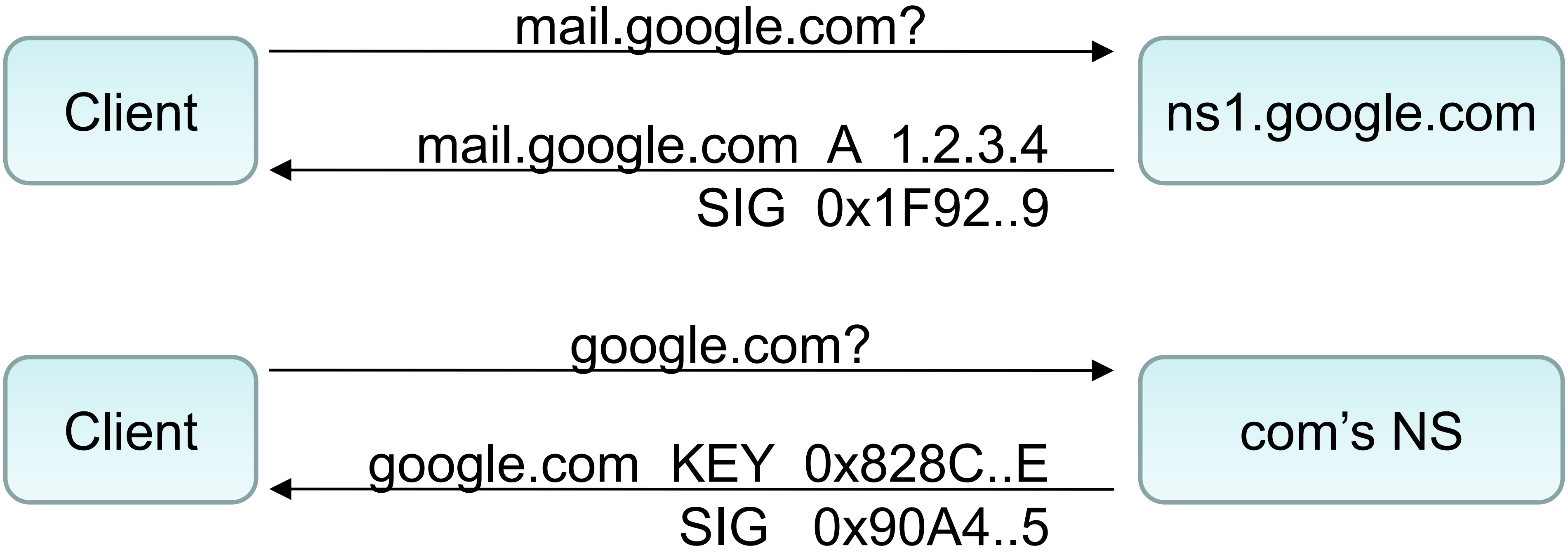
Client

mail.google.com?

mail.google.com A 1.2.3.4

ns1.google.com

DNSSEC:



Client

mail.google.com?

mail.google.com A 1.2.3.4
SIG 0x1F92..9

ns1.google.com

Client

google.com?

google.com KEY 0x828C..E
SIG 0x90A4..5

com’s NS

Issues With DNSSEC ?

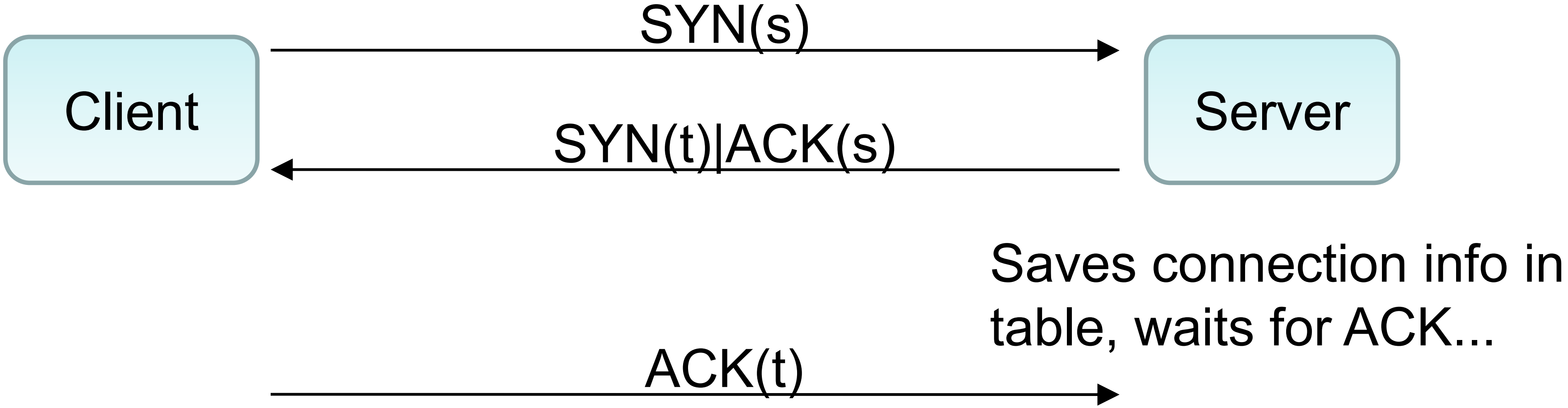
- Issue #1: Replies are Big
  - E.g., query for "`berkeley.edu`" returns 1400+ bytes
  - DoS amplification
  - Increased latency on low-capacity links
  - Headaches w/ older libraries that assume replies < 512B

- Issue #2: *Partial deployment*
  - Suppose `.com` not signing, though `google.com` is
  - Major practical concern. What do we do?
  - Can wire additional key into resolver (doesn't scale)
  - Or: outsource to trusted third party ("lookaside")
    - Wire their key into resolver, they sign numerous early adopters

Issues With DNSSEC, con’t

- Issue #3: *Partial deployment*
  - What do you do with unsigned/unvalidated results?
  - If you trust them, weakens incentive to upgrade
  - If you don’t trust them, a whole lot of things break
- Issue #4: Negative results (“no such name”)
  - What statement does the nameserver sign?
  - If “`gabluph.google.com`” doesn’t exist, then have to do dynamic key-signing (expensive) for any bogus request
    - DoS vulnerability
  - Instead, sign (off-line) statements about order of names
    - E.g., sign “`gabby.google.com` followed by `gabrunk.google.com`”
    - Thus, can see that `gabluph.google.com` can’t exist
  - But: now attacker can enumerate all names that exist :-(

TCP handshake



Client

Server

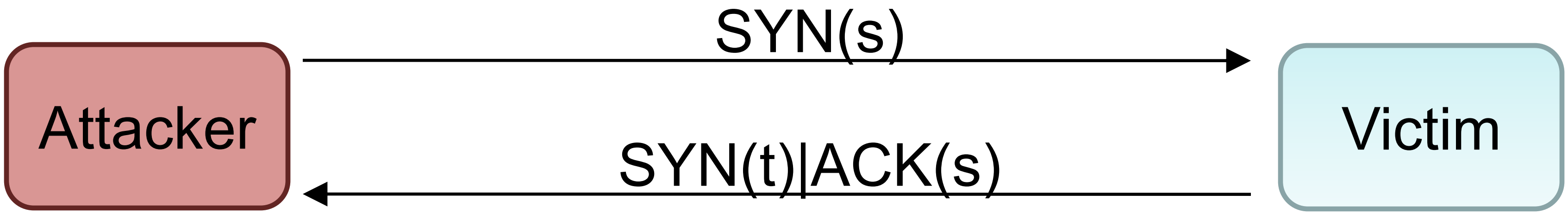
SYN(s)

SYN(t)|ACK(s)

Saves connection info in table, waits for ACK...

ACK(t)

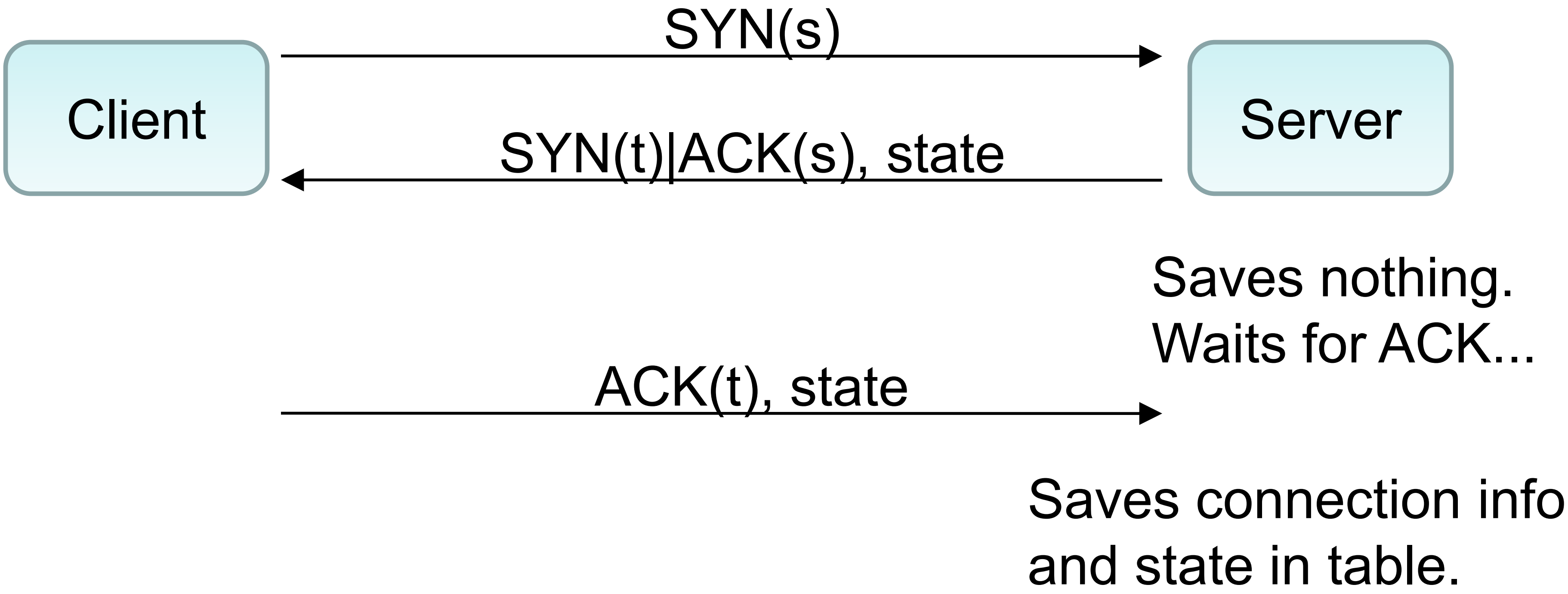
SYN flooding attack



Saves connection info in table, waits for ACK...

Attacker repeats this until Victim’s table is full.

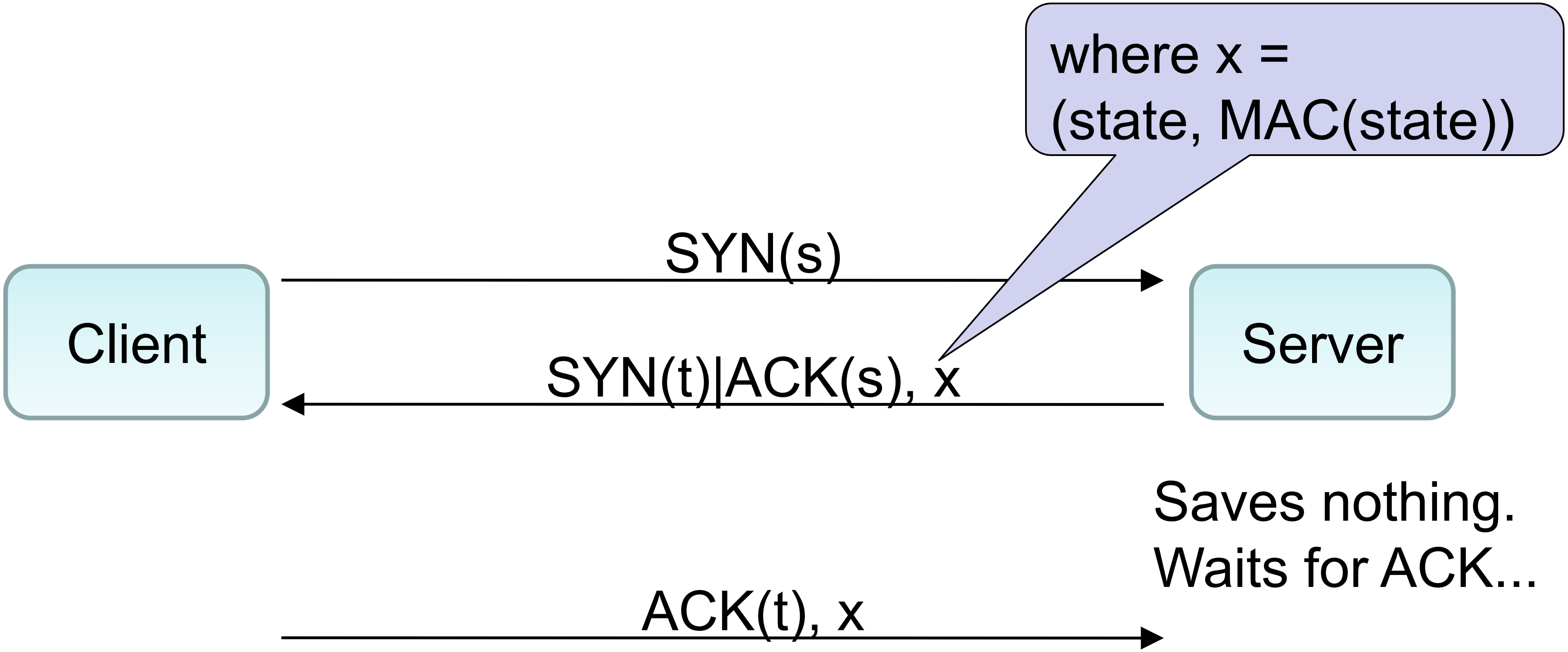
SYN cookies (naive)



Saves nothing.
Waits for ACK...

Saves connection info
and state in table.

SYN cookies (simplified)



where x =
(state, MAC(state))

Client

Server

SYN(s)

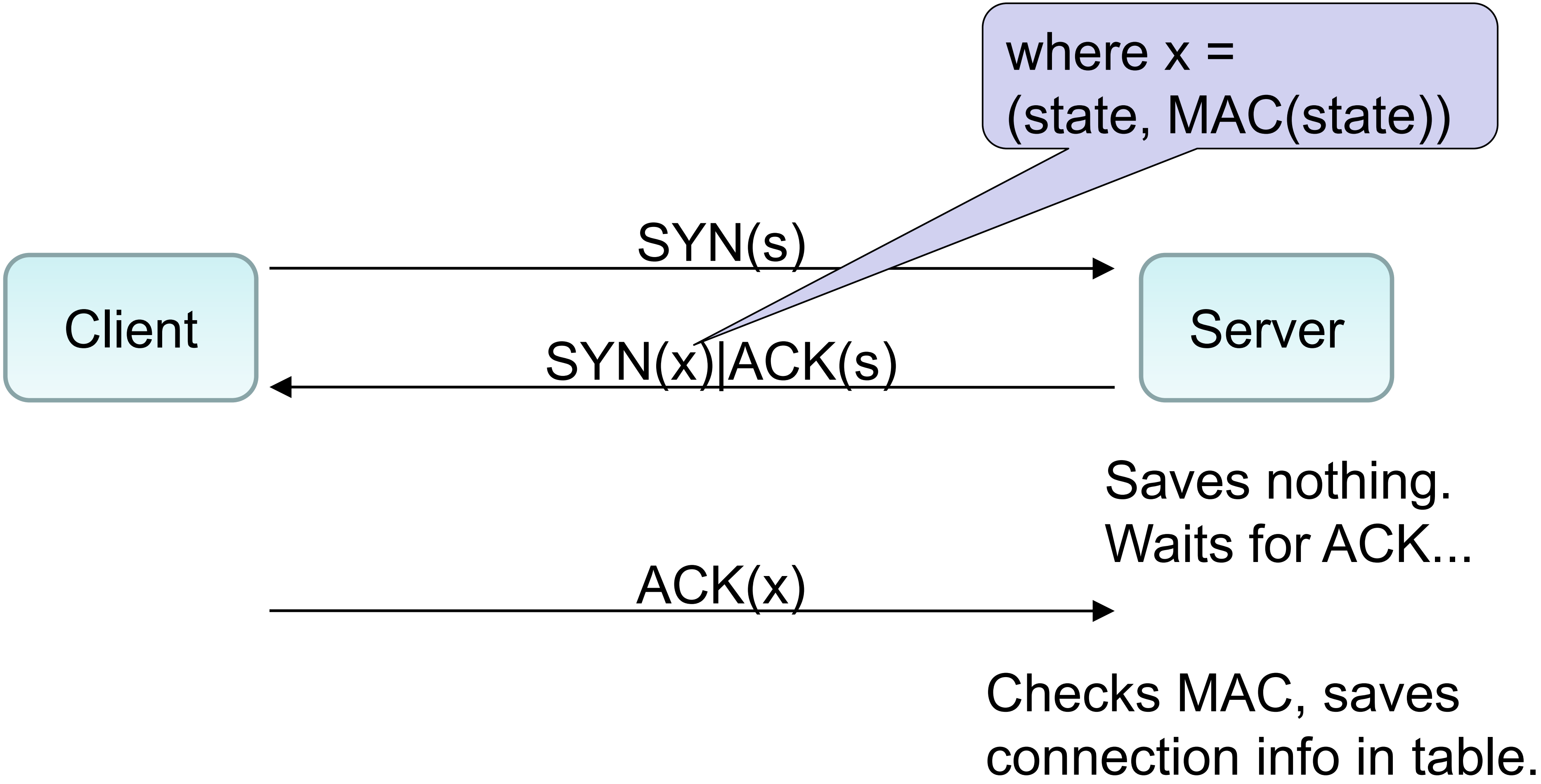
SYN(t)|ACK(s), x

Saves nothing.
Waits for ACK...

ACK(t), x

Checks MAC, saves
connection info in table.

SYN cookies (actual)



where x =
(state, MAC(state))

Client

Server

SYN(s)

SYN(x)|ACK(s)

Saves nothing.
Waits for ACK...

ACK(x)

Checks MAC, saves
connection info in table.