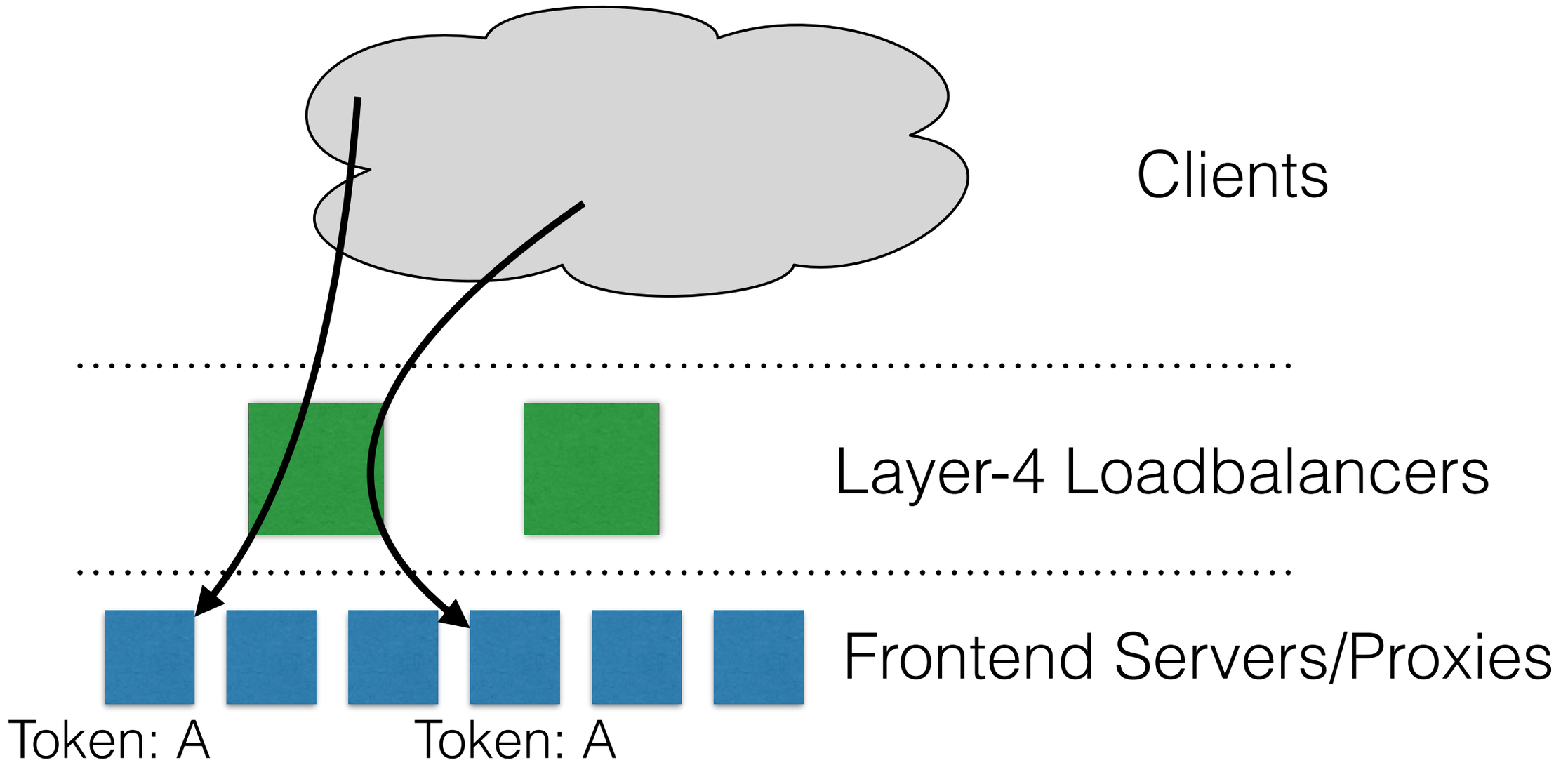
Rethinking the MPTCP handshake

Christoph Paasch <cpaasch@apple.com>

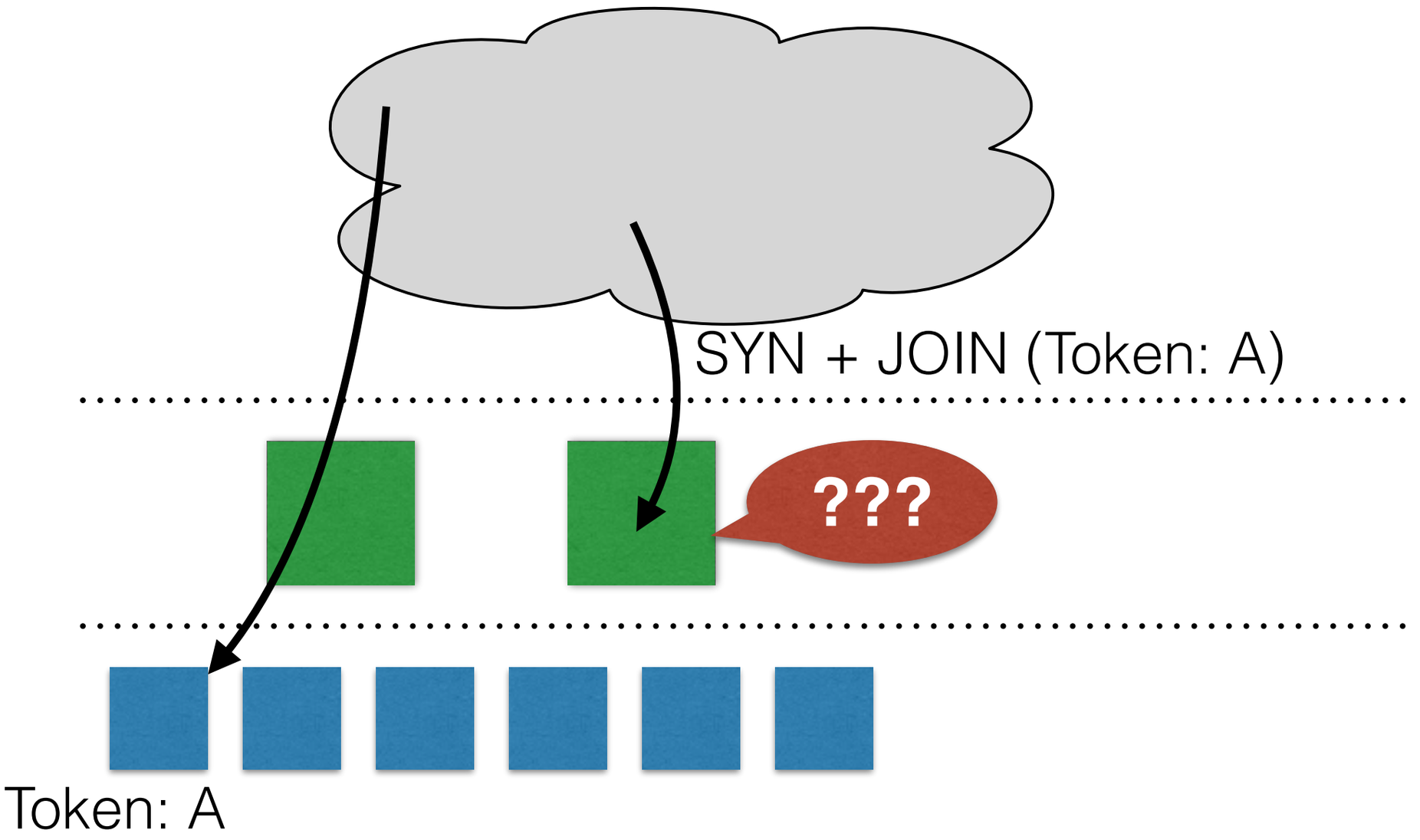
Weaknesses of the current handshake

Deployment behind layer-4 loadbalancers



Different servers may choose the same token/key

Deployment behind layer-4 loadbalancers



Different loadbalancers are not aware of the MPTCP-state

Deployment behind layer-4 loadbalancers

- Deployment behind a loadbalancer is very difficult
  - ➡ Not possible to do classic layer-4 loadbalancer
  - ➡ Thus, requires unicast IP on each server, implying DNS-based load balancing
  - ➡ Scalability becomes a major concern

Security: Different attacker models per subflow

- Initial subflow
    - Attacker **cannot** eavesdrop the SYNs
        - ➡ MPTCP sends keys in plaintext
- Additional subflows
    - Attacker **can** eavesdrop the SYNs
        - ➡ Must use HMAC to prove knowledge of keys without revealing them

Security

- Inconsistent attacker models on the MPTCP subflows
- Security-sensitive applications will anyways rely on TLS (or equivalent)

What can we change?

RFC6824-bis will bump the version number

➡ Opportunity to address these challenges!

Rethinking the handshake

MPTCP behind loadbalancers:

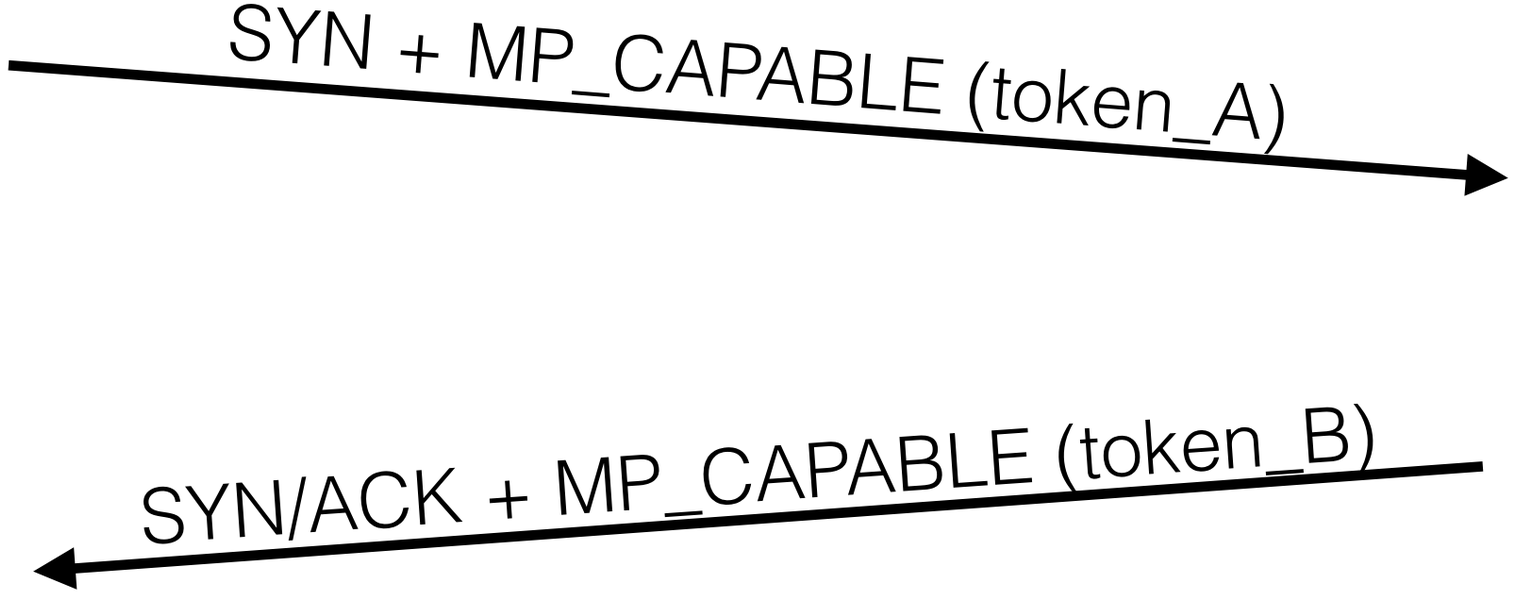
- Token should be locally “meaningful”

Security aspects of MPTCP

- Consistent attacker models across all subflows
- Leverage higher-layer security for MPTCP

“Design MPTCP for tomorrow’s protocol stack: HTTP/2, TLS, MPTCP, IPv6” - O. Bonaventure

Making the token locally “meaningful”



Token announced explicitly, makes it locally “meaningful” on the server-side

✓ Loadbalancers are supported

Tomorrow's protocol stack: HTTP/2, TLS, MPTCP, IPv6

- Do we need a separate key-negotiation mechanism for MPTCP, when TLS already does it?

- Security provided by TLS is superior to the one MPTCP can ever provide

- Use a derivate of the TLS-key for MPTCP's HMAC
  (cfr., draft-paasch-mptcp-ssl & draft-bonaventure-mptcp-tls)

Tentative proposed handshake

SYN + MP_CAPABLE (token_A, key_selection)

SYN/ACK + MP_CAPABLE (token_B, key_selection)

ACK + MP_CAPABLE (token_A, token_B, key_selection)

key_selection to choose among a set of key-negotiation techniques (e.g., TLS, PSK, null-Key,…)

Conclusion

- As RFC6824-bis bumps the version number, we have an opportunity to address a lot of issues
- Loadbalancer-support is key for widespread deployment
- We can address the security issues as well by leveraging TLS (is a push for TLS as well)