P4 VPN Authentication

Authentication of VPN Traffic on a Network Device with P4

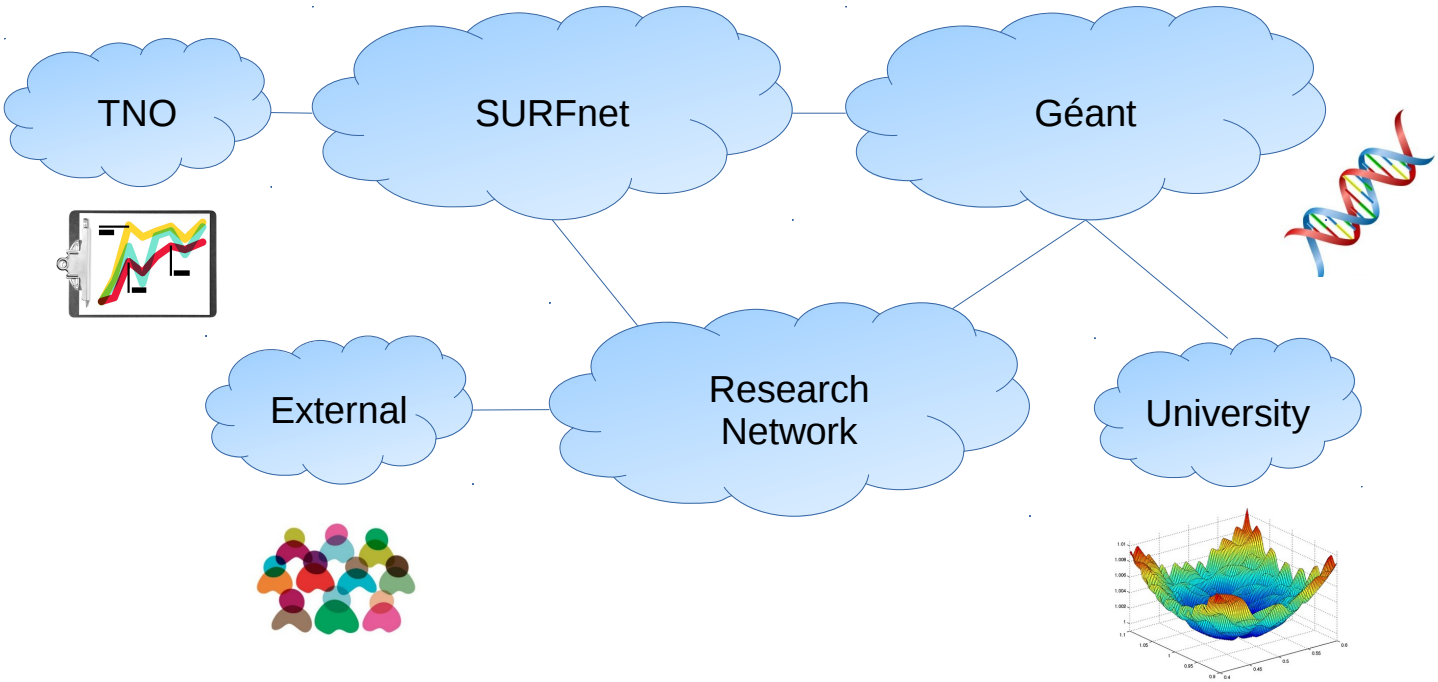
Jeroen Klomp

University of Amsterdam
System and Network Engineering
Research Project

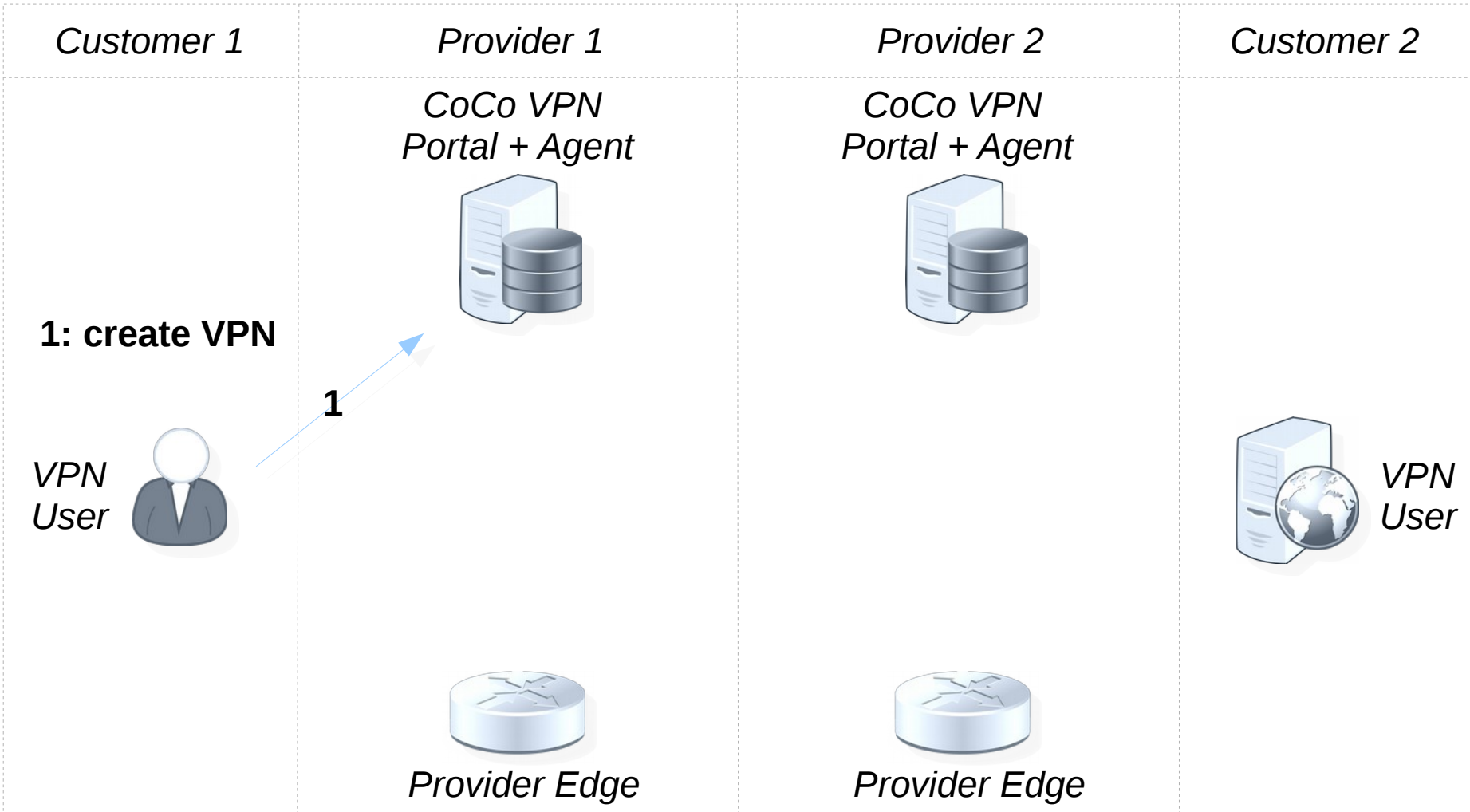
June 29, 2016

CoCo Introduction

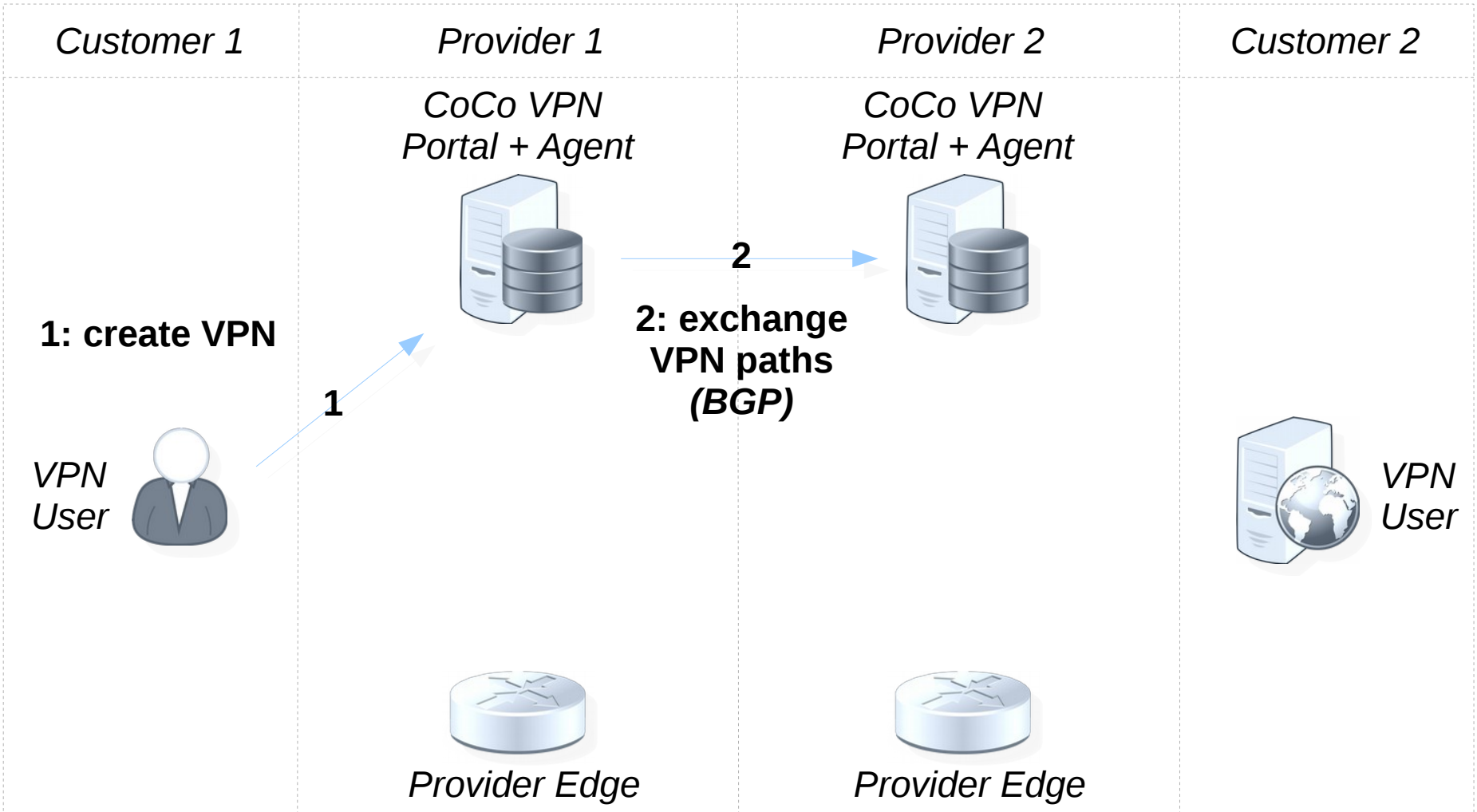
- Community Connection (CoCo)
  - User-initiated multi-domain VPN service
  - Support eScience
  - Prototype phase; no proper authentication



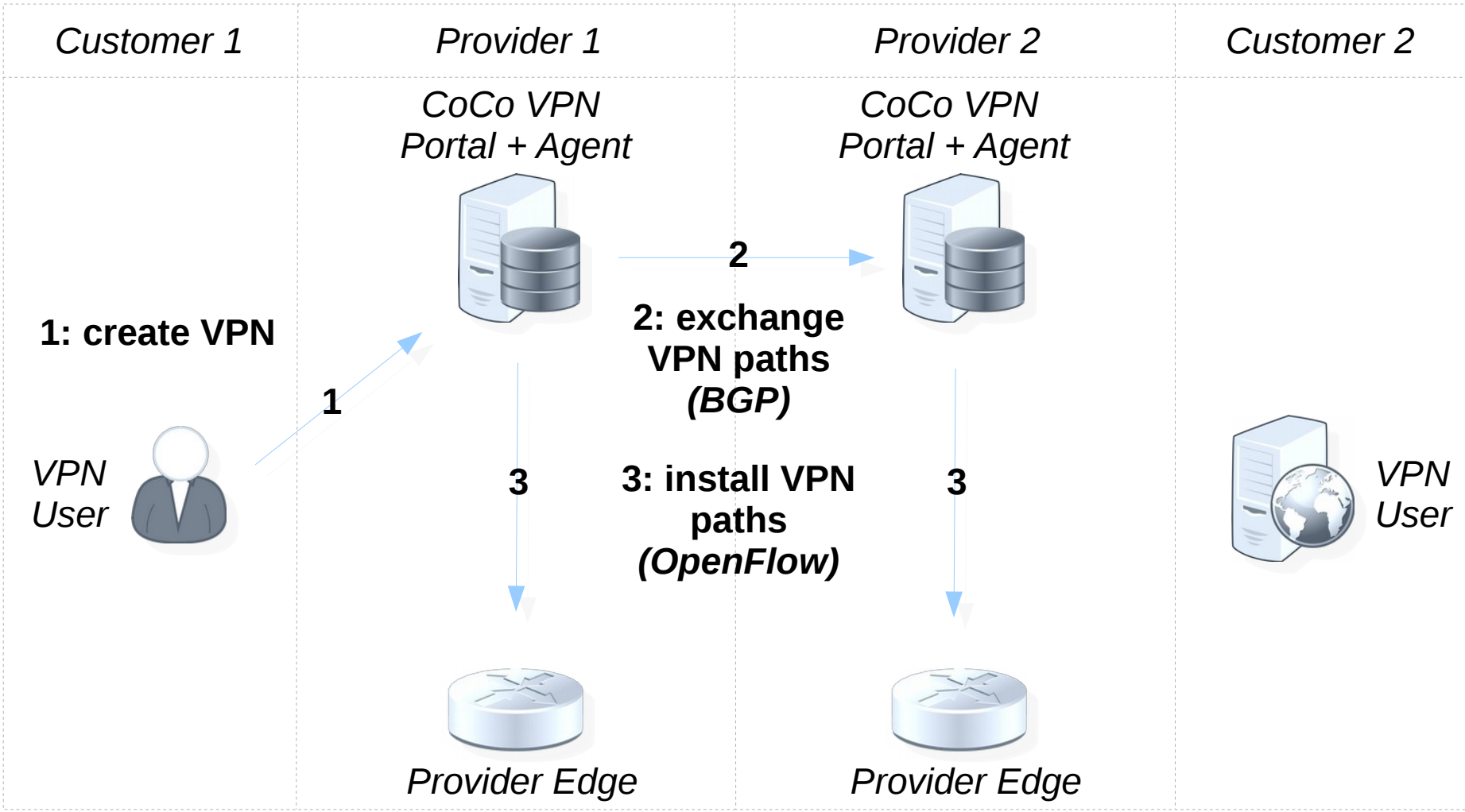
CoCo Overview



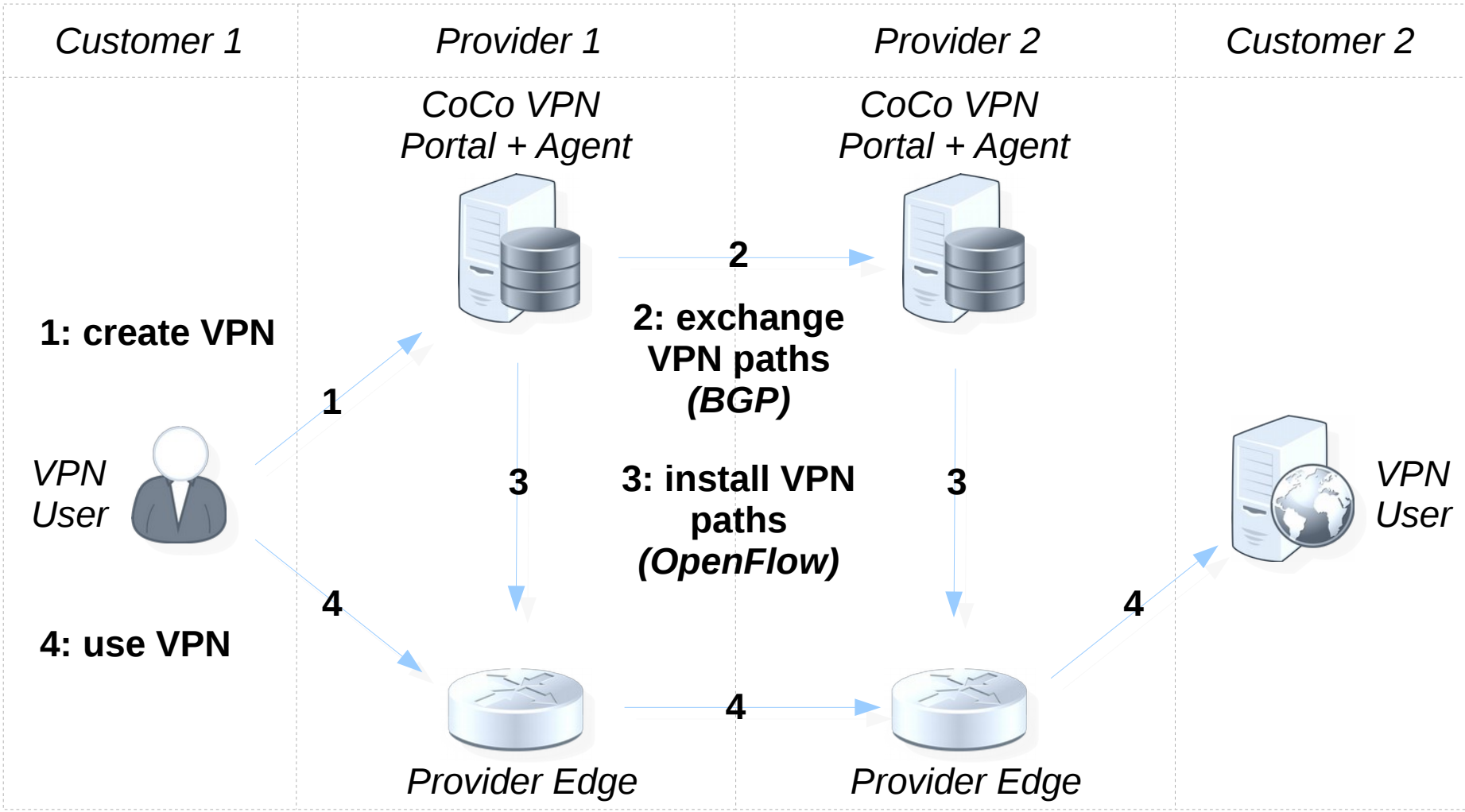
CoCo Overview



CoCo Overview

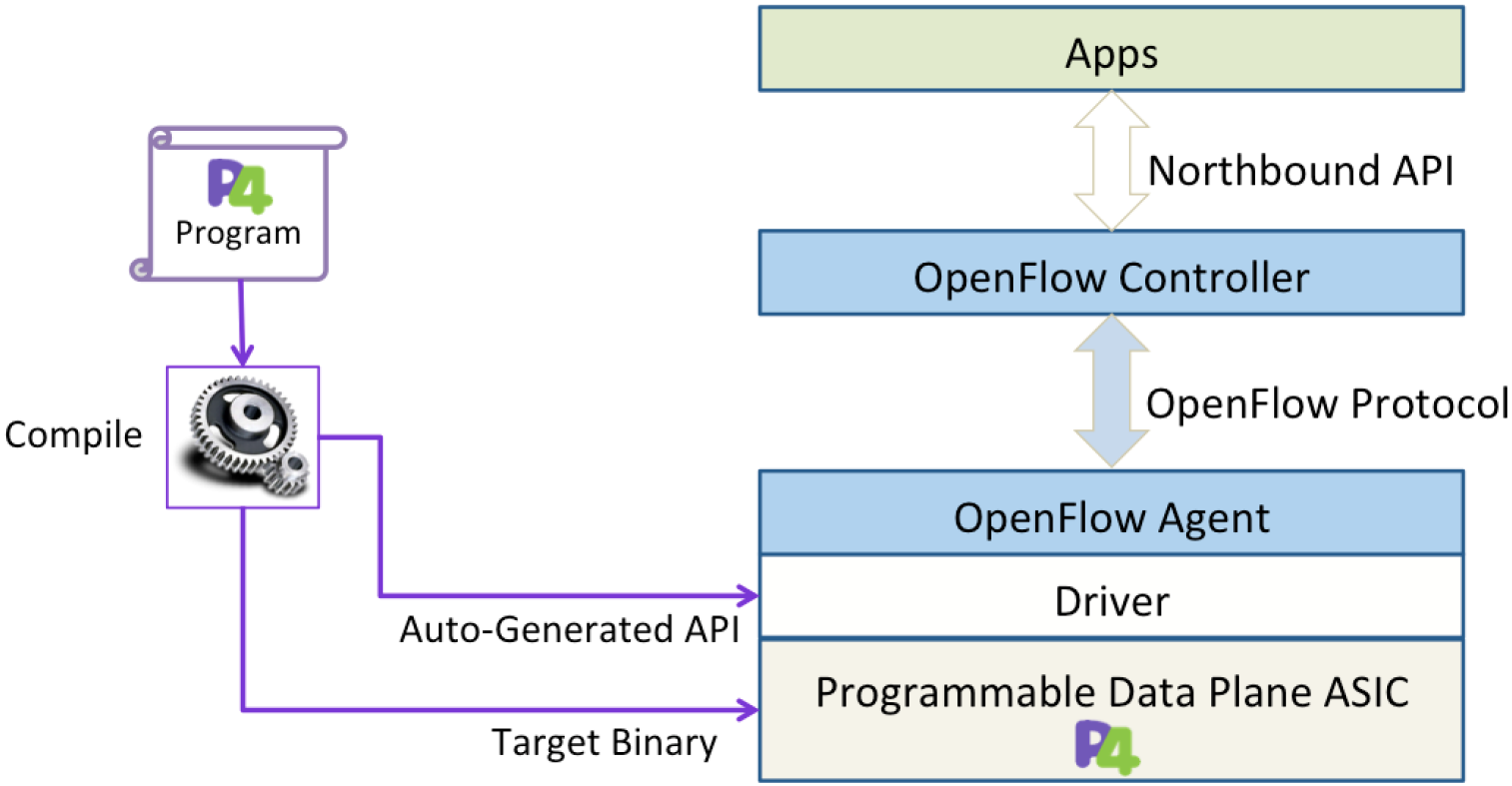


CoCo Overview

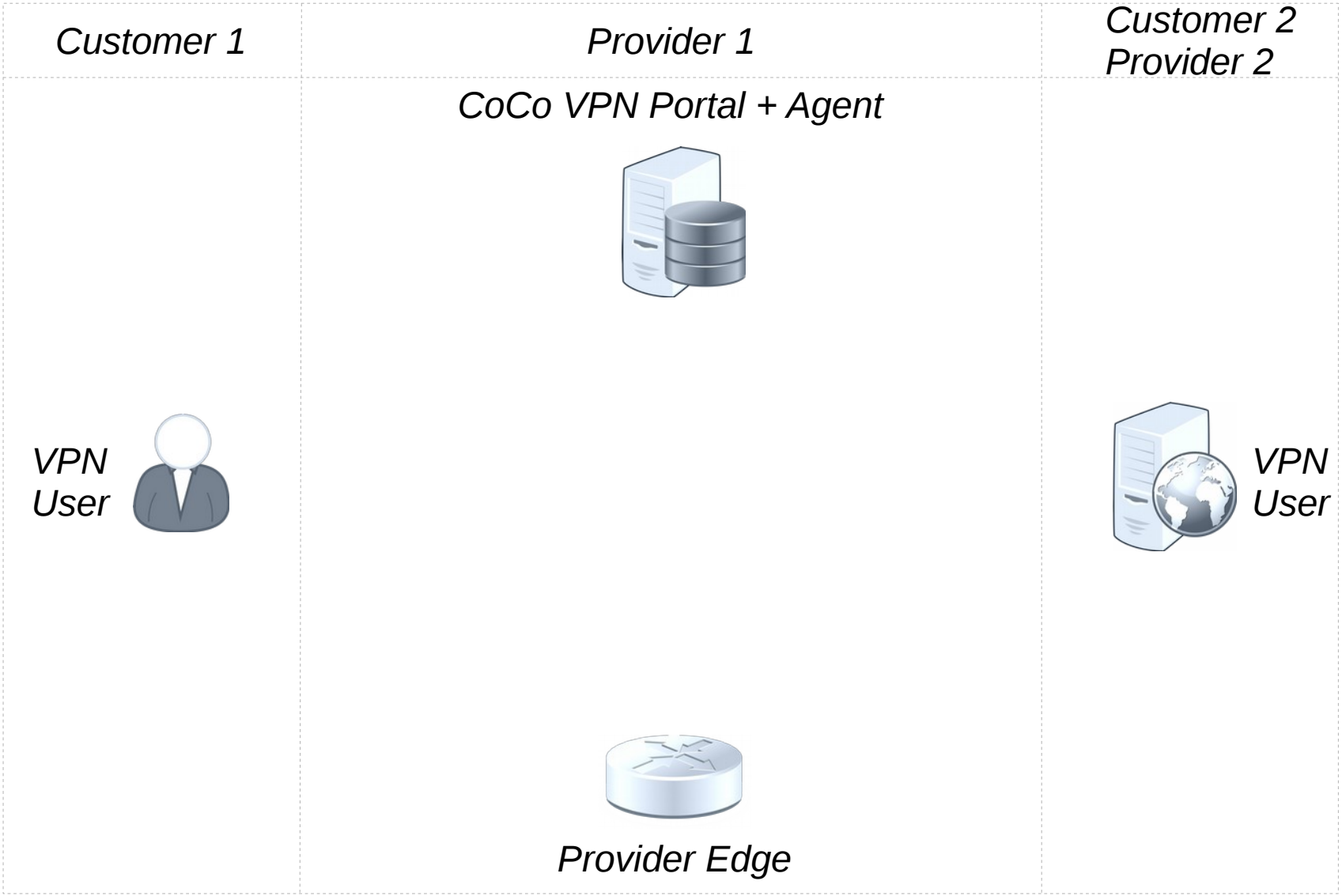


P4 Overview

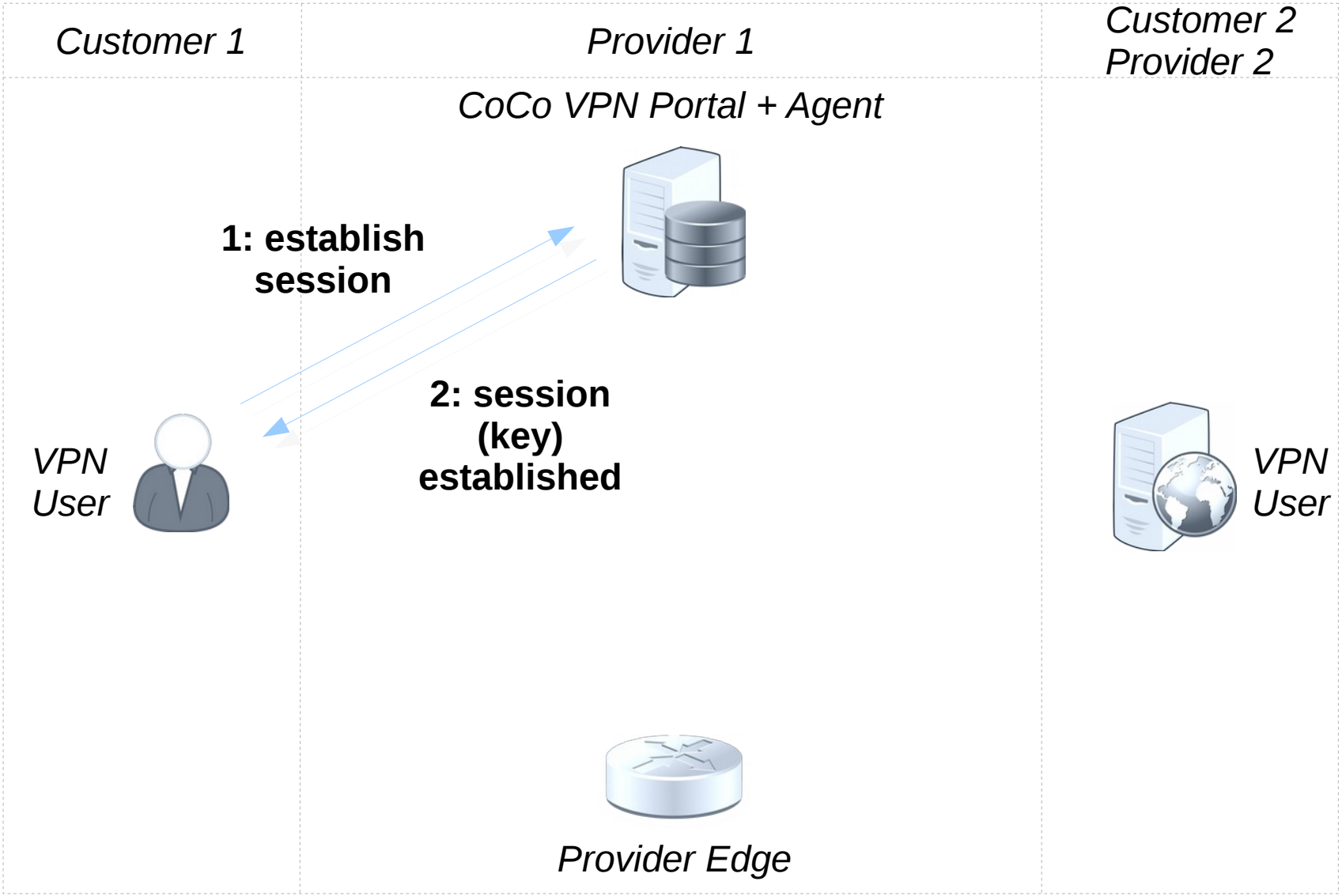
P4 & OpenFlow



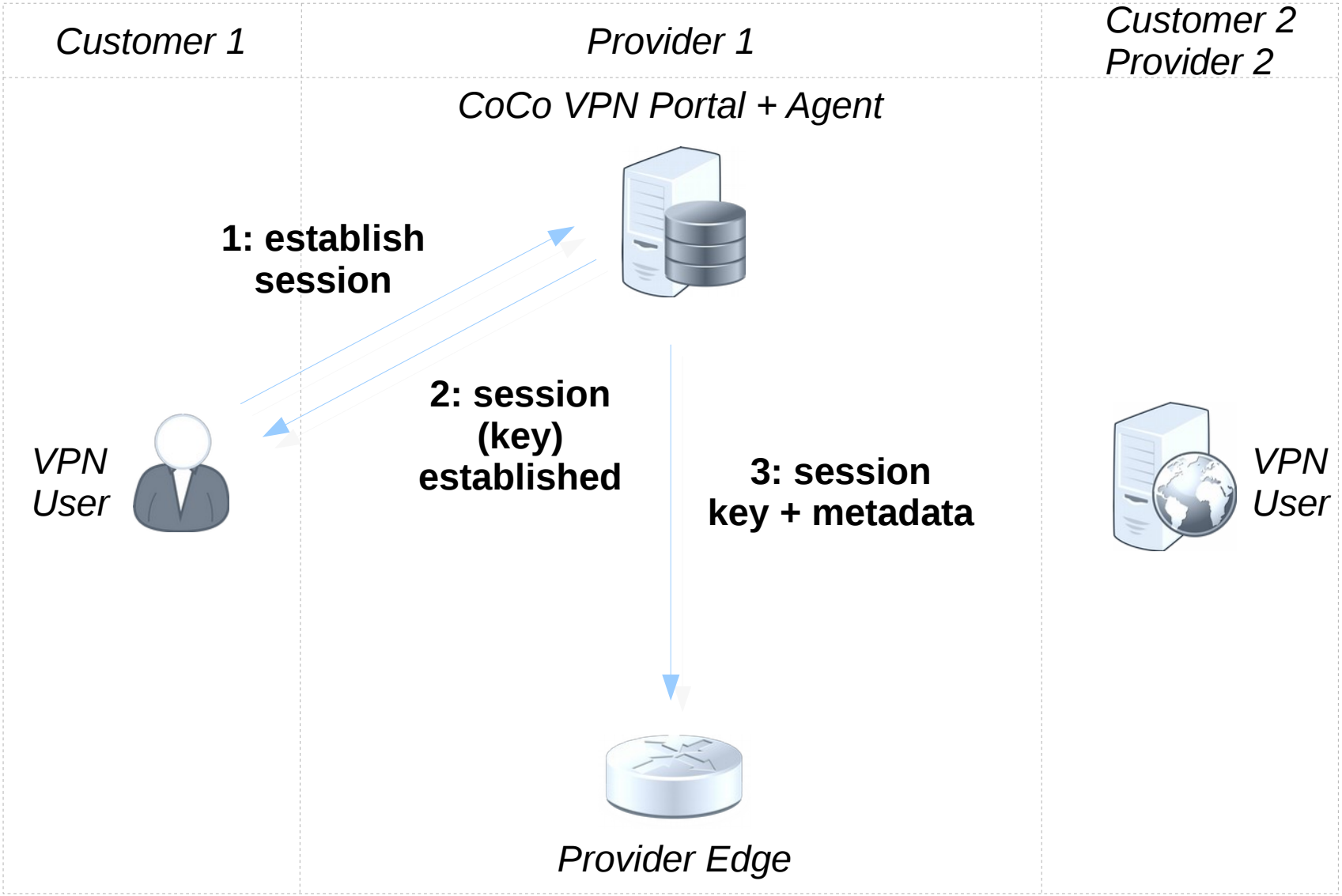
Authentication Use Case



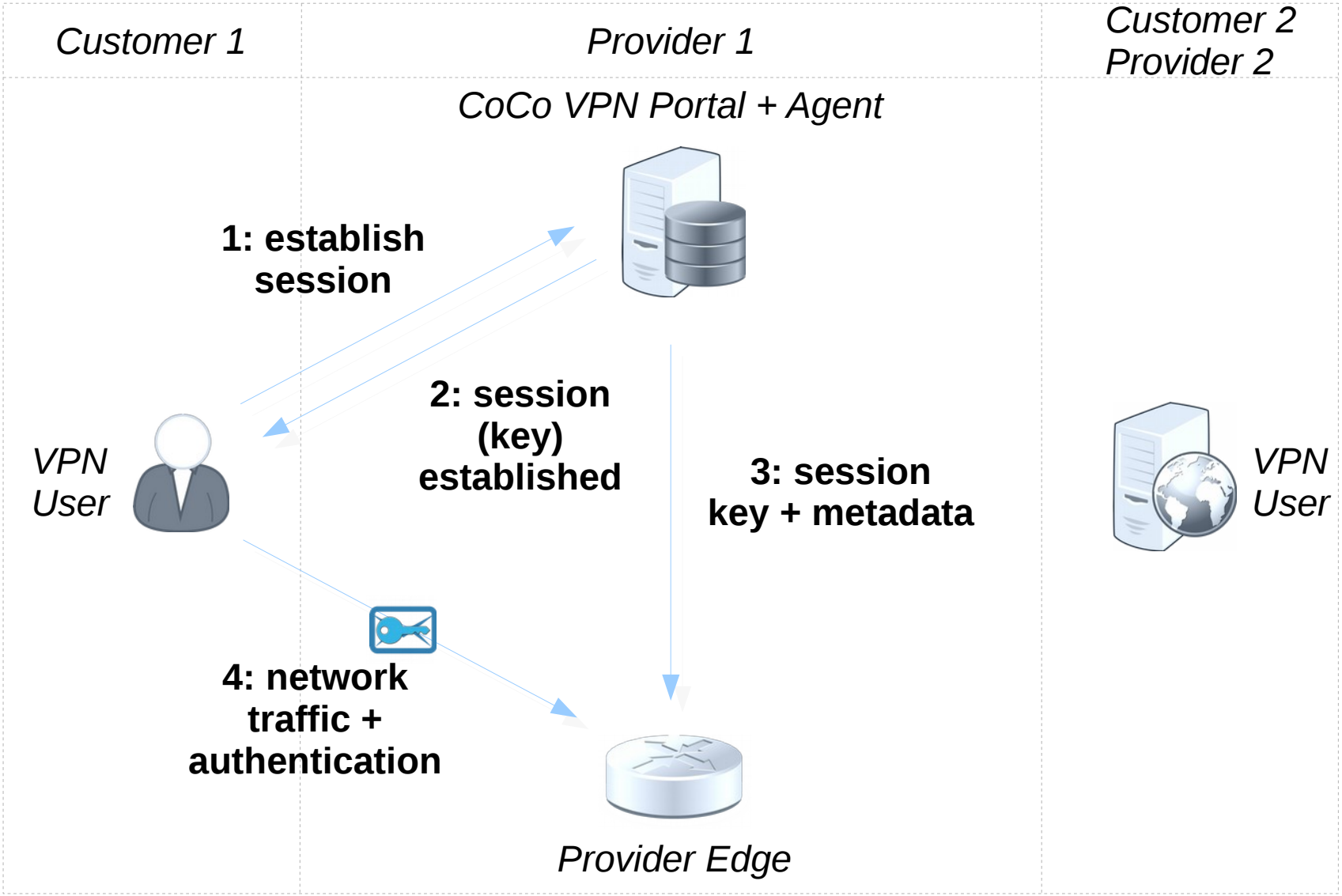
Authentication Use Case



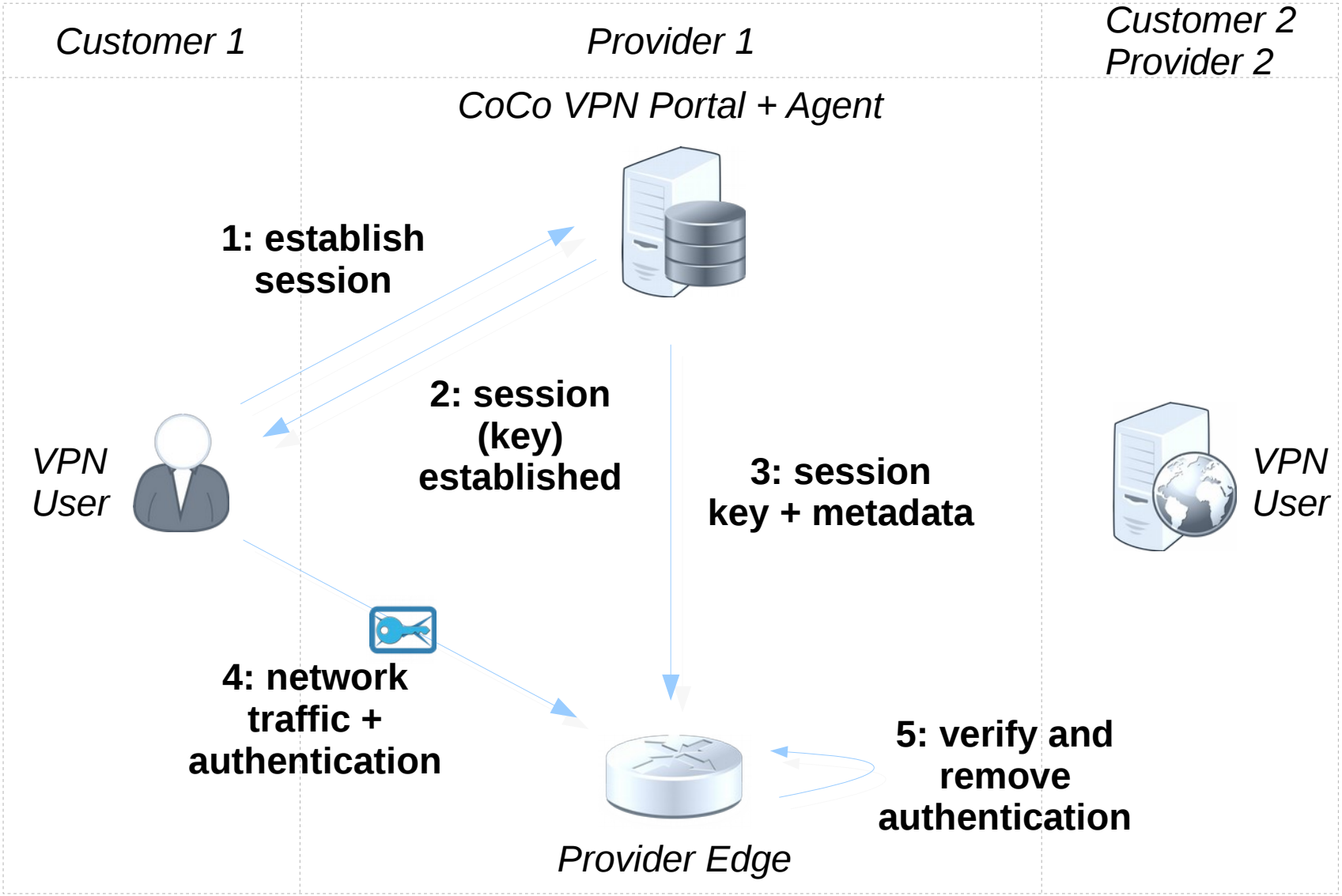
Authentication Use Case



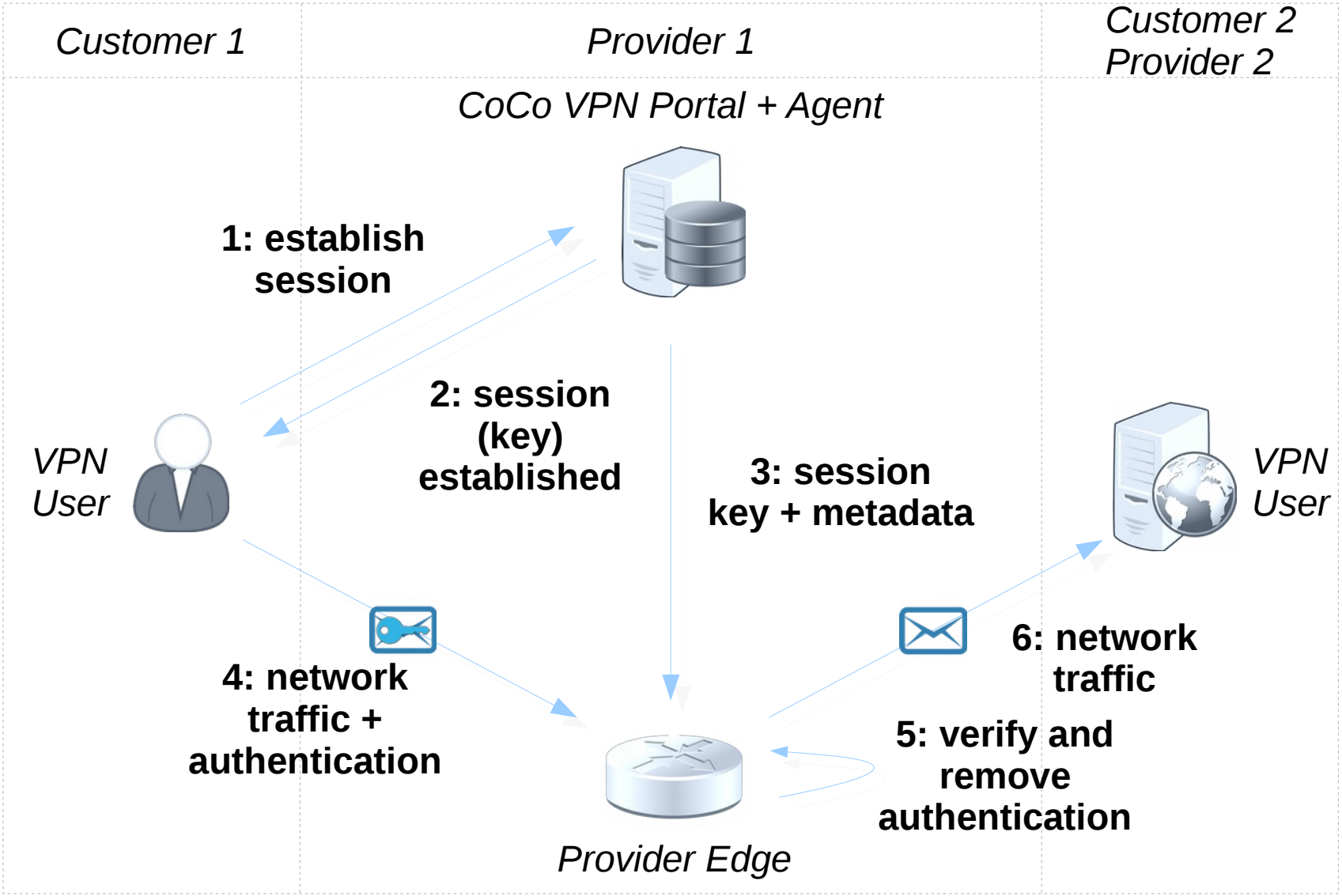
Authentication Use Case



Authentication Use Case



Authentication Use Case



Authentication Scheme

- Requirements
  - Secure
  - Support research

Authentication Scheme

- Requirements
  - Secure
  - Support research
- Authentication method
  - Message authentication code (MAC)

Authentication Scheme

- Requirements
  - Secure
  - Support research
- Authentication method
  - Message authentication code (MAC)
  - Like checksums, MAC algorithms cannot be implemented in P4 itself
    - Use external function provided by the target
    - Possibly a standard library with MAC algorithms?
    - Language needs facility for key input

Authentication Scheme

- Requirements
  - Secure
  - Support research
- Authentication method
  - Message authentication code (MAC)
  - Like checksums, MAC algorithms cannot be implemented in P4 itself
    - Use external function provided by the target
    - Possibly a standard library with MAC algorithms?
    - Language needs facility for key input
- Authentication protocol

Authentication Protocol

- IPSEC Authentication Header?
  - Contains all necessary fields
    - Security Parameters Index (SPI): Security Association → session ID
    - Sequence number → replay protection
    - Integrity Check Value (ICV) → variable length MAC

Offset		Type	0								1								2								3							
Bit	Octet		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	AH	Next Header								Payload Length								Reserved															
32	4		**Security Parameters Index (SPI)**: *session identifier*																															
64	8		**Sequence Number**: *replay protection*																															
96	12		**Integrity Check Value (ICV)** *(variable MAC)*																															
128	16																																	
160	20																																	
192	24																																	
224	28																																	
256	32		*[~ padding]*																															
288	36	ICMP/ UDP/ TCP/...	*Transport protocol and payload*																															

Implementation in P4

- Distinguish sessions
  - Session identifier table containing session IDs and session keys

Implementation in P4

- Distinguish sessions
  - Session identifier table containing session IDs and session keys
- Sequence number
  - Register per session

Implementation in P4

- Distinguish sessions
  - Session identifier table containing session IDs and session keys
- Sequence number
  - Register per session
- MAC
  - Session key mixed with message
  - Hash[1] calculated and stored as metadata via primitive action:
    `MODIFY_FIELD_WITH_HASH_BASED_OFFSET()`

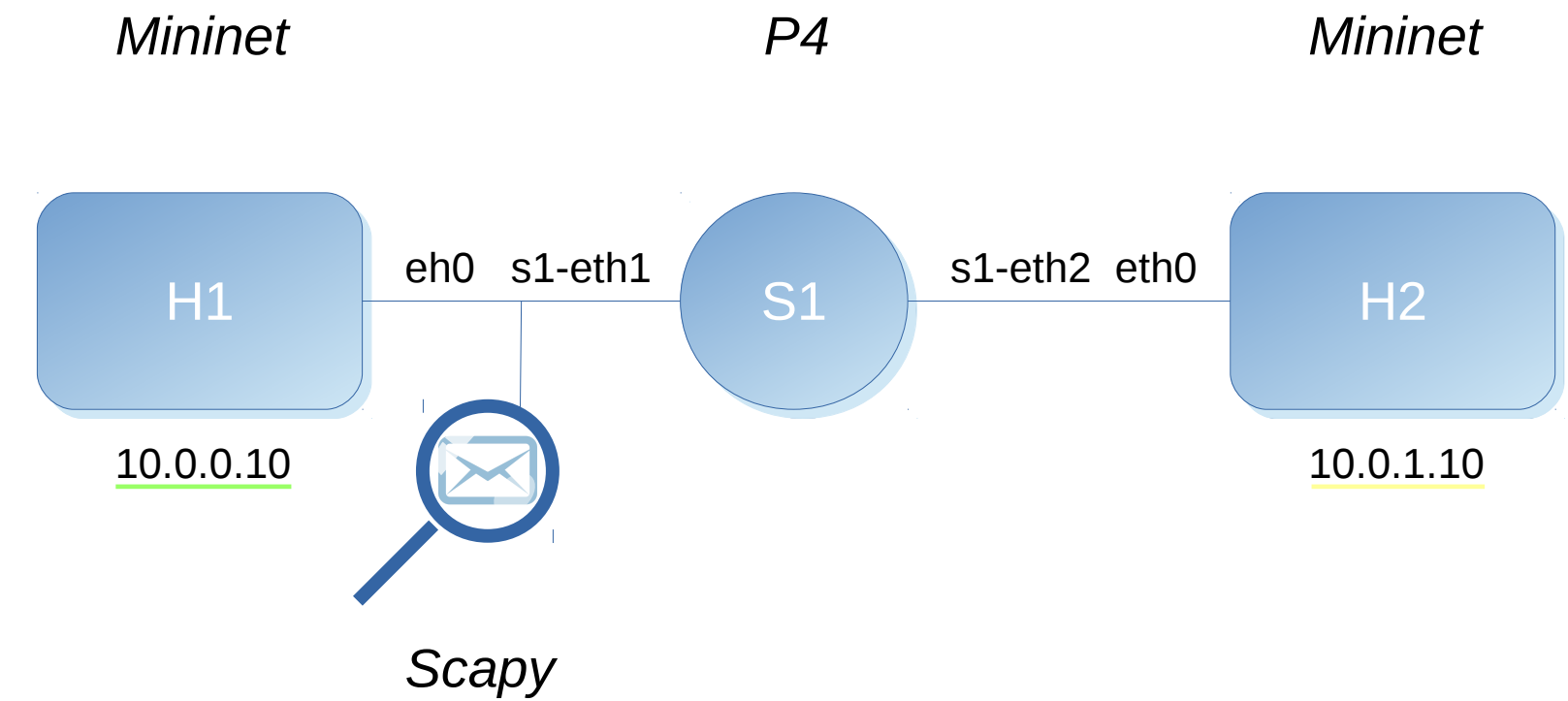
[1] 'simulated' via checksum

Simplified Authentication Protocol

- GRE (Generic Routing Encapsulation)
  - Has necessary fields
    - Key: session ID
    - Sequence number
    - Checksum: (mis)used for MAC simulation
  - Easily craft packets e.g., via Scapy

Offset		Type	0								1								2								3							
Bit	Octet		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	GRE	Flags: *CKS*				Reserved0									Version			Protocol Type: *0000 (possibly GRE keepalive)*														
32	4		**Checksum**: *MAC (CRC16)*																**Offset**: *key (not on wire)*														
64	8		**Key**: *session identifier*																															
96	12		**Sequence Number**																															
128	16	ICMP	*ICMP echo request with random payload*																															

Test Setup


Mininet
P4
Mininet
H1
eh0
s1-eth1
S1
s1-eth2
eth0
H2
10.0.0.10
10.0.1.10
Scapy
GRE(key)/ICMP(rqst)/'123'
ICMP(rqst)/'123'
ICMP(rply)/'123'
ICMP(rply)/'123'

Demonstration

Results

- Concepts work[2]
  - Packets accepted only with correct key
  - Sequence number correctly checked & updated
  - Multiple session IDs and keys supported simultaneously

[2] although using an extremely weak form of MAC

Results

- Concepts work[2]
  - Packets accepted only with correct key
  - Sequence number correctly checked & updated
  - Multiple session IDs and keys supported simultaneously
- P4 language and software targets still work in progress
  - Problems with dropping traffic
  - Register operations not yet in specification
  - Key length supported?

[2] although using an extremely weak form of MAC

Conclusion

- Authentication with P4 is feasible
  - But requires new P4 features and target support
  - Keep authentication scheme & P4 program simple

Conclusion

- Authentication with P4 is feasible
  - But requires new P4 features and target support
  - Keep authentication scheme & P4 program simple
- Lots of caveats
  - Target limitations; cryptographic algorithms; NAT; IP fragmentation; packet forwarding mode (cut-through); sequence number synchronisation; asymmetric flows

Conclusion

- Authentication with P4 is feasible
  - But requires new P4 features and target support
  - Keep authentication scheme & P4 program simple
- Lots of caveats
  - Target limitations; cryptographic algorithms; NAT; IP fragmentation; packet forwarding mode (cut-through); sequence number synchronisation; asymmetric flows
- Where to go from here?
  - Add cryptographic means to P4
  - Further design CoCo architecture & authentication scheme
    - Implement in P4, controller & client
    - End-to-end authentication & encryption?

Questions?