SOMMARIO

- Introduzione a webrtc
- Concetti base VOIP
- Comunicazione peer-to-peer
- Demo peerconnection
- C++ in stile webrtc
- Esempio datachannel
- Architettura videoconferenza

YURI VALENTINI

- SW Windows e Linux
- Linguaggi: C/C++, C#, Python
- Videoconferenza e VOIP
- yuroller@gmail.com
- https://github.com/yuroller

- Web Real-Time Communication
- browser, mobile-phones, IoT
- BSD license
- No brevetti

STORIA

- 2010 Google acquisisce GIPS ($68.2 mil)
- 2011 Google rilascia WebRTC
- IETF standard protocolli
- W3C standard api browser
- 2013 video call fra browser diversi
- 2014 OpenWebRTC di Ericsson Research
- 2017 WebRTC 1.0 Candidate Recommendation

SUPPORTO

- PC (Chrome, Firefox, Edge, Safari)
- Android (Chrome, Firefox)
- IOS 11 (MobileSafari/WebKit)
- ...

API JAVASCRIPT

- getUserMedia: accesso ai dispositivi
- RTCPeerConnection: audio/video tra peer
    - elaborazione segnali
    - codec
    - comunicazione fra peer
    - sicurezza
    - gestione banda
- RTCDataChannel: trasferimento dati
- getStats: ottenere statistiche

CODEC

- obbligatori
  - PCMA/PCMU, Opus
  - DTMF via Telephone Event
- altri di Google WebRTC
  - ISAC, G722, ILBC
  - CN (Confort Noise)
  - VP8, VP9
- codec esterni: es. H264

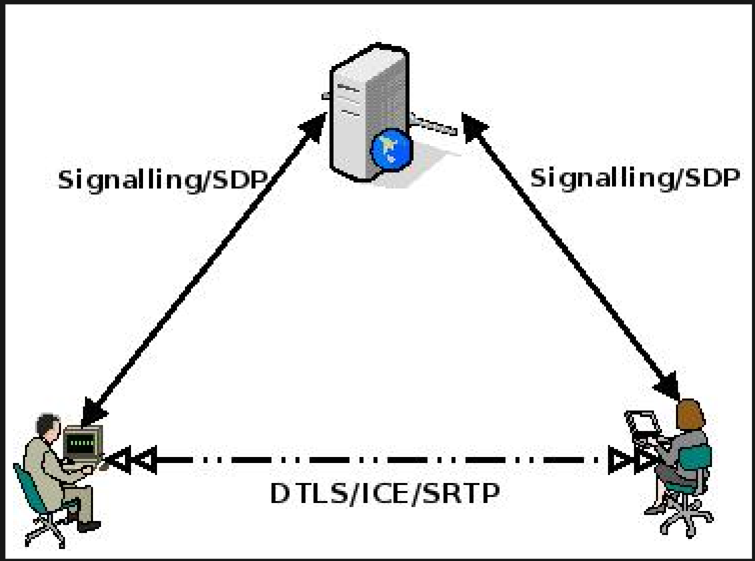
SDP

- descrive i media
- negoziazione offerta/risposta
- utilizzato in VOIP SIP
- sdp anatomy
- "alternativa" ORTC

TERMINOLOGIA VOIP/WEBRTC

- RTP, RTCP
- DTLS, SRTP
- STUN (Session Traversal Utilities for NAT)
- TURN (Traversal Using Relays around NAT)
- ICE (Interactive Connectivity Establishment)

DEMO PEERCONNECTION



INSTALLAZIONE SU WINDOWS

scaricare e scompattare depot_tools

```
c:\progetti>PATH=C:\progetti\depot_tools;%PATH%
c:\progetti>set DEPOT_TOOLS_WIN_TOOLCHAIN=0
c:\progetti>gclient
c:\progetti\webrtc>fetch webrtc
c:\progetti\webrtc\src>gclient sync
c:\progetti\webrtc\src>gn gen out\Release --ide=vs \
  --args="is_clang=false is_debug=false rtc_include_tests=false"
c:\progetti\webrtc\src>ninja -C out\Release
```

#INCLUDE

```
api and subdirectories
common_audio/include
media/base
media/engine
modules/audio_coding/include
modules/audio_device/include
modules/audio_processing/include
modules/bitrate_controller/include
modules/congestion_controller/include
modules/include
modules/remote_bitrate_estimator/include
modules/rtp_rtcp/include
modules/rtp_rtcp/source
modules/utility/include
modules/video_coding/codecs/h264/include
modules/video_coding/codecs/vp8/include
modules/video_coding/codecs/vp9/include
modules/video_coding/include
pc, rtc_base, system_wrappers/include
```

GOOGLE C++

- C++11 (abseil)
- exception sono vietate
- uso limitato degli stream stl
- template semplici

RTC::THREAD

- implementa rtc::MessageQueue
- inviare e ricevere messaggi (Post(), Get(), ..)
- socket server
- eseguire codice su thread

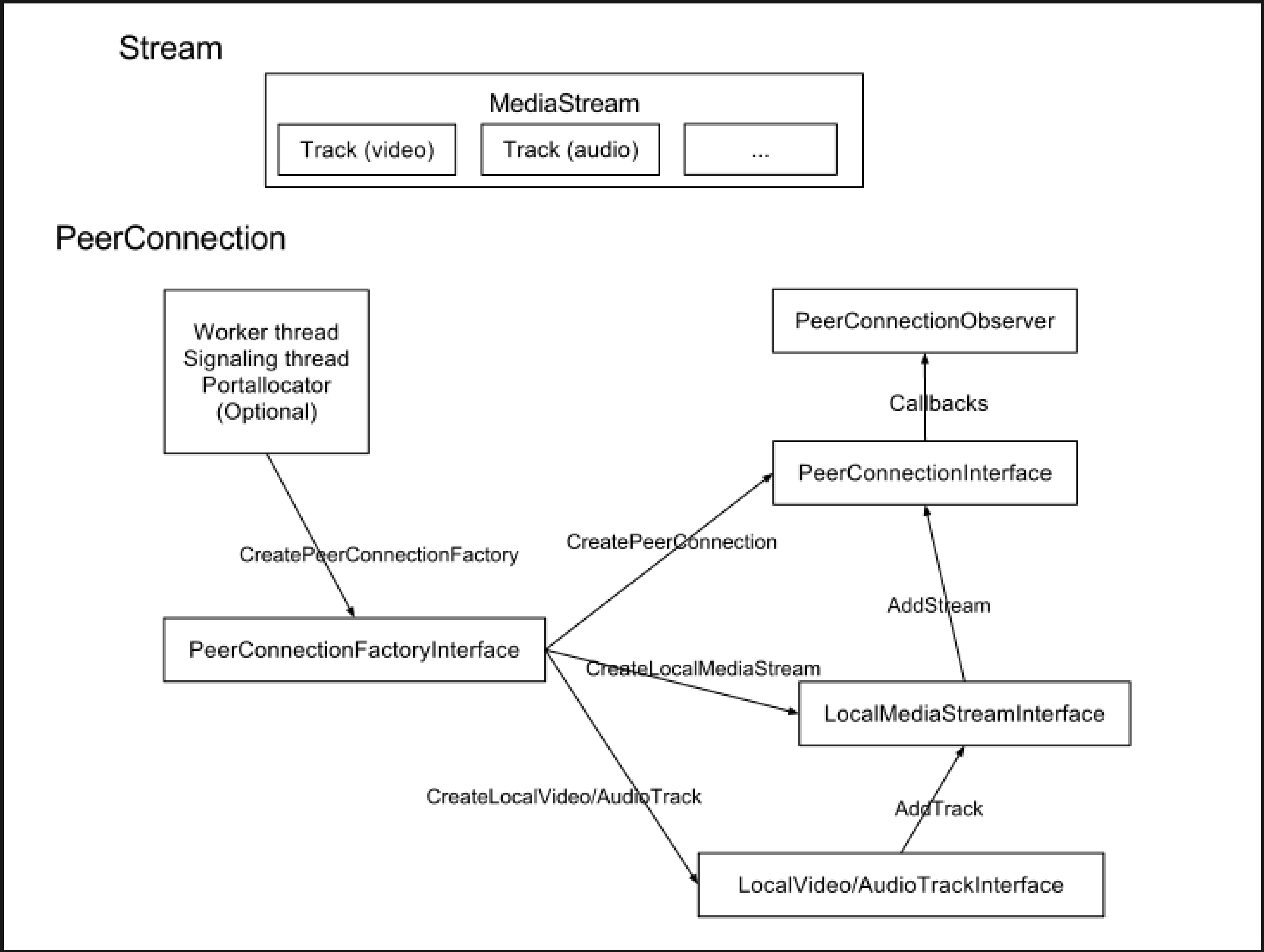
RTC::SCOPED_REFPTR

- reference count interno
- tracciabilità di oggetti reference counted (es. COM)
- stessa memoria fra T e rtc::scoped_refptr<T>
- meno allocazioni/deallocazioni
- località di memoria

OBSERVER WEBRTC

- esito di operazione asincrone
- eventi asincroni
- eseguito da thread specifico
- gestione memoria semplificata

OGGETTI WEBRTC



PEERCONNECTION 1

```
class PeerConnectionInterface : public rtc::RefCountInterface
public:
  virtual rtc::scoped_refptr<StreamCollectionInterface>
    local_streams() = 0;
  virtual rtc::scoped_refptr<StreamCollectionInterface>
    remote_streams() = 0;
  virtual bool AddStream(MediaStreamInterface* stream) = 0;
  virtual void RemoveStream(MediaStreamInterface* stream) = 0;
  virtual RTCErrorOr<rtc::scoped_refptr<RtpSenderInterface>>
    AddTrack(
      rtc::scoped_refptr<MediaStreamTrackInterface> track,
      const std::vector<std::string>& stream_ids);
  virtual bool RemoveTrack(RtpSenderInterface* sender);
...
```

PEERCONNECTION 2

```
...
  virtual void GetStats(
    rtc::scoped_refptr<RtpSenderInterface> selector,
    rtc::scoped_refptr<RTCStatsCollectorCallback> callback) {}
  virtual void GetStats(
    rtc::scoped_refptr<RtpReceiverInterface> selector,
        rtc::scoped_refptr<RTCStatsCollectorCallback> callback

  virtual rtc::scoped_refptr<DataChannelInterface>
    CreateDataChannel(
      const std::string& label,
      const DataChannelInit* config) = 0;
...
```

PEERCONNECTION 3

```
...
  virtual void CreateOffer(
    CreateSessionDescriptionObserver* observer,
    const RTCOfferAnswerOptions& options) = 0;
  virtual void CreateAnswer(
    CreateSessionDescriptionObserver* observer,
    const RTCOfferAnswerOptions& options) = 0;
  virtual void SetLocalDescription(
    SetSessionDescriptionObserver* observer,
    SessionDescriptionInterface* desc) = 0;
  virtual void SetRemoteDescription(
    SetSessionDescriptionObserver* observer,
    SessionDescriptionInterface* desc) {}
...
```

PEERCONNECTION 4

```
...
  virtual bool AddIceCandidate(
    const IceCandidateInterface* candidate) = 0;
  virtual void Close() = 0;
};
```

PEERCONNECTIONOBSERVER 1

```
class PeerConnectionObserver {
public:
  virtual ~PeerConnectionObserver() = default;

  virtual void OnSignalingChange(
    PeerConnectionInterface::SignalingState new_state) = 0;

  virtual void OnAddStream(
    rtc::scoped_refptr<MediaStreamInterface> stream) {}
  virtual void OnRemoveStream(
                rtc::scoped_refptr<MediaStreamInterface> strea
  virtual void OnDataChannel(
    rtc::scoped_refptr<DataChannelInterface> data_channel) = 0
...
```

PEERCONNECTIONOBSERVER 2

```
...
  virtual void OnRenegotiationNeeded() = 0;
  virtual void OnIceConnectionChange(
      PeerConnectionInterface::IceConnectionState new_state) =
  virtual void OnConnectionChange(
      PeerConnectionInterface::PeerConnectionState new_state)
  virtual void OnIceGatheringChange(
      PeerConnectionInterface::IceGatheringState new_state) =
  virtual void OnIceCandidate(
    const IceCandidateInterface* candidate) = 0;
...
```

PEERCONNECTIONOBSERVER 3

```
...
  virtual void OnAddTrack(
    rtc::scoped_refptr<RtpReceiverInterface> receiver,
    const std::vector<rtc::scoped_refptr<MediaStreamInterface>
      streams) {}

  virtual void OnRemoveTrack(
    rtc::scoped_refptr<RtpReceiverInterface> receiver) {}
};
```

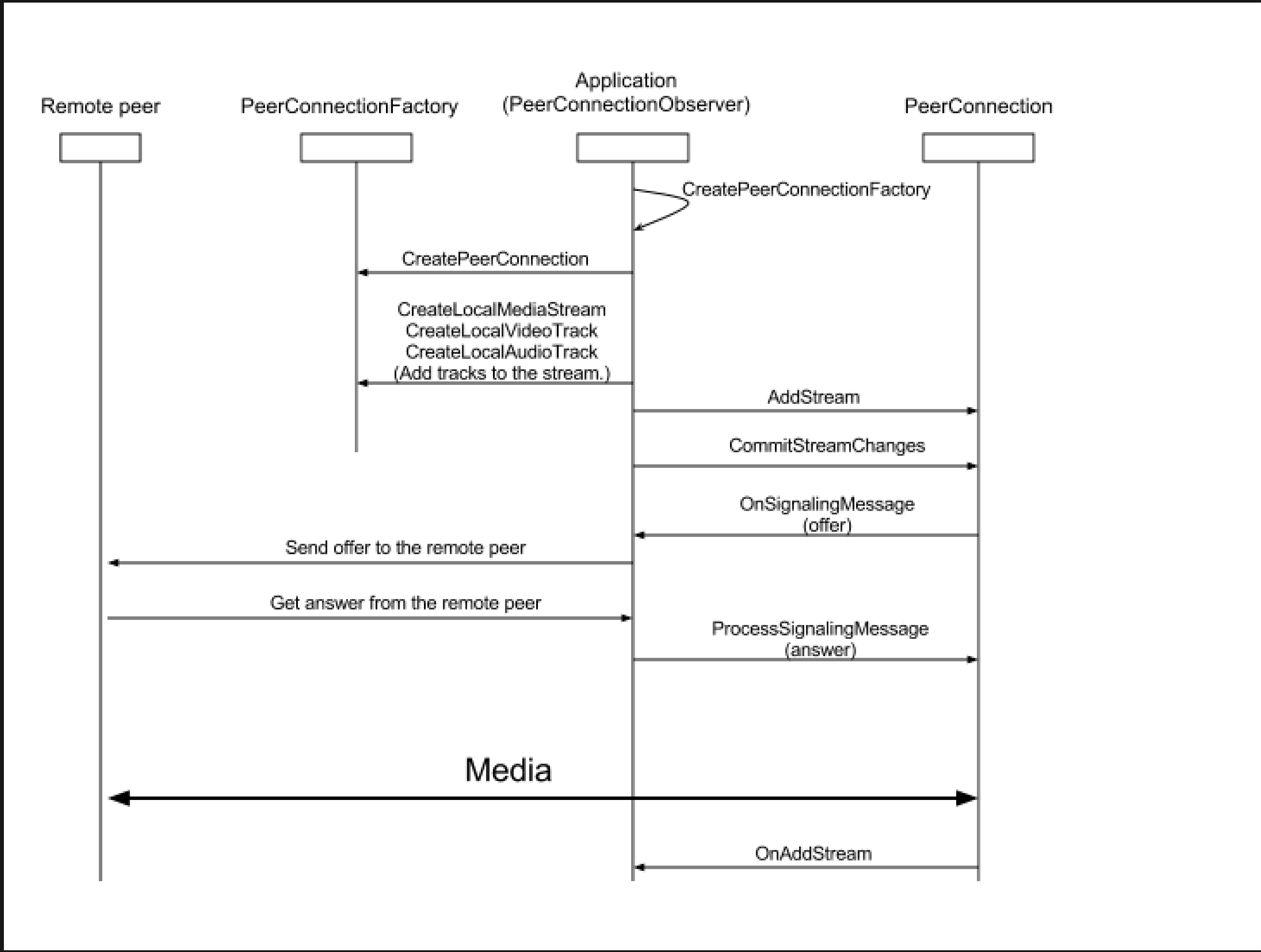
MEDIASTREAM 1

```
class MediaStreamInterface : public rtc::RefCountInterface,
     public NotifierInterface {
public:
  virtual std::string id() const = 0;

  virtual AudioTrackVector GetAudioTracks() = 0;
  virtual VideoTrackVector GetVideoTracks() = 0;
  virtual rtc::scoped_refptr<AudioTrackInterface> FindAudioTra
     const std::string& track_id) = 0;
  virtual rtc::scoped_refptr<VideoTrackInterface> FindVideoTra
          const std::string& track_id) = 0;
...
```

MEDIASTREAM 2

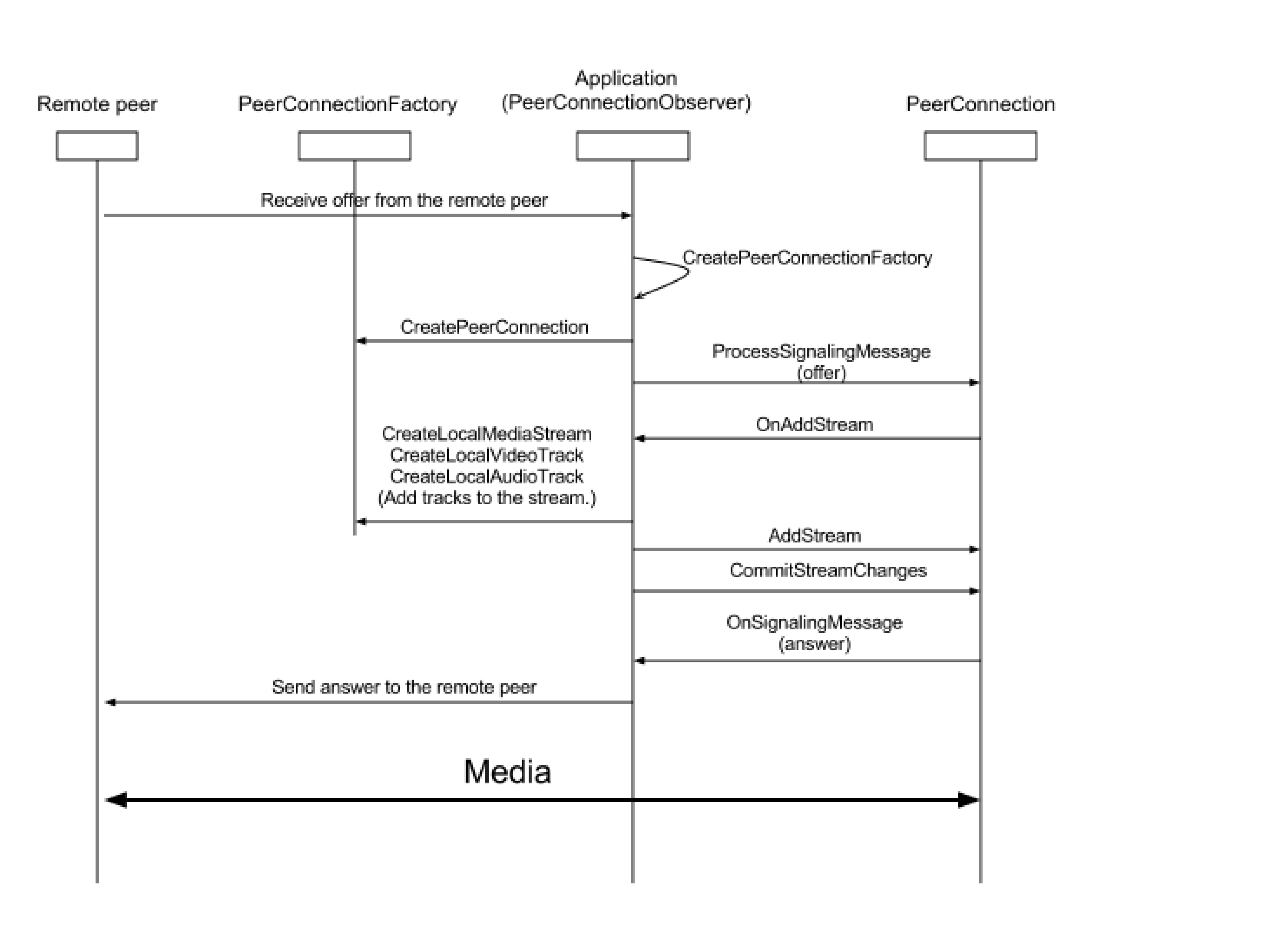
```
...
  virtual bool AddTrack(AudioTrackInterface* track) = 0;
  virtual bool AddTrack(VideoTrackInterface* track) = 0;
  virtual bool RemoveTrack(AudioTrackInterface* track) = 0;
  virtual bool RemoveTrack(VideoTrackInterface* track) = 0;

protected:
  ~MediaStreamInterface() override = default;
};
```

SET UP CALL

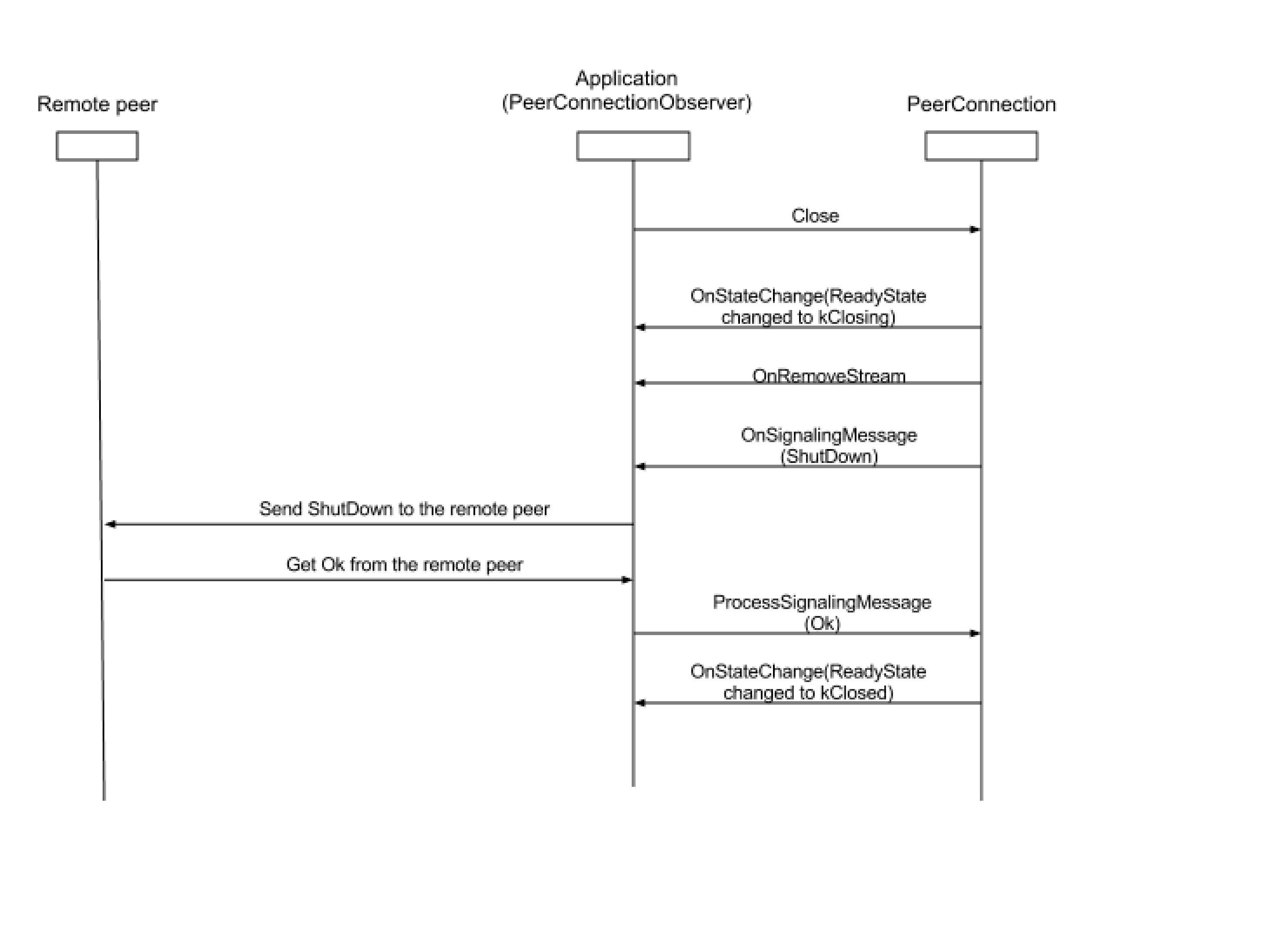


Remote peer
PeerConnectionFactory
Application
(PeerConnectionObserver)
PeerConnection
CreatePeerConnectionFactory
CreatePeerConnection
CreateLocalMediaStream
CreateLocalVideoTrack
CreateLocalAudioTrack
(Add tracks to the stream.)
AddStream
CommitStreamChanges
OnSignalingMessage
(offer)
Send offer to the remote peer
Get answer from the remote peer
ProcessSignalingMessage
(answer)
Media
OnAddStream

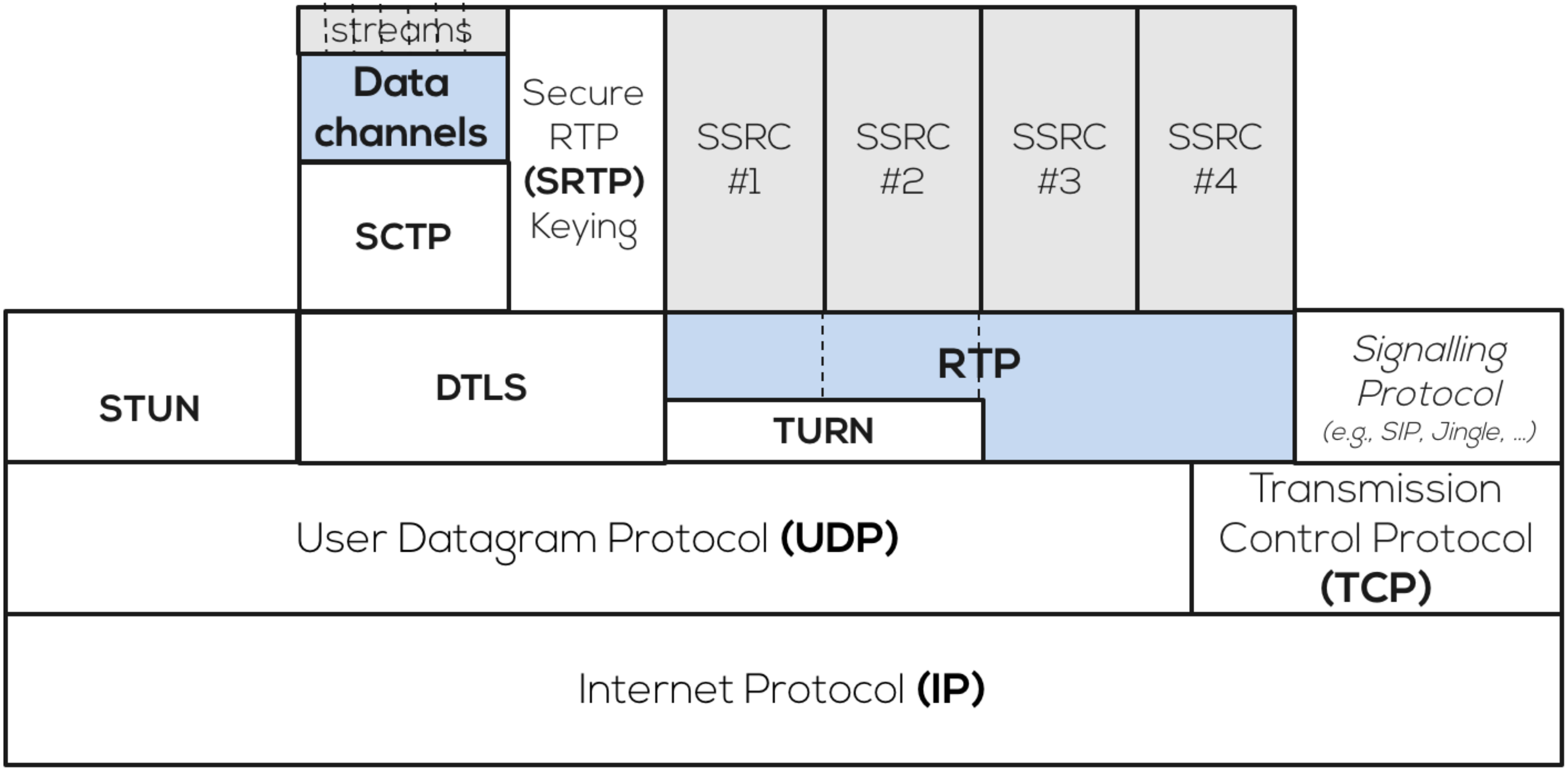
RECEIVE CALL



CLOSE CALL


Remote peer
Application
(PeerConnectionObserver)
PeerConnection
Close
OnStateChange(ReadyState
changed to kClosing)
OnRemoveStream
OnSignalingMessage
(ShutDown)
Send ShutDown to the remote peer
Get Ok from the remote peer
ProcessSignalingMessage
(Ok)
OnStateChange(ReadyState
changed to kClosed)

DEMO DATACHANNEL

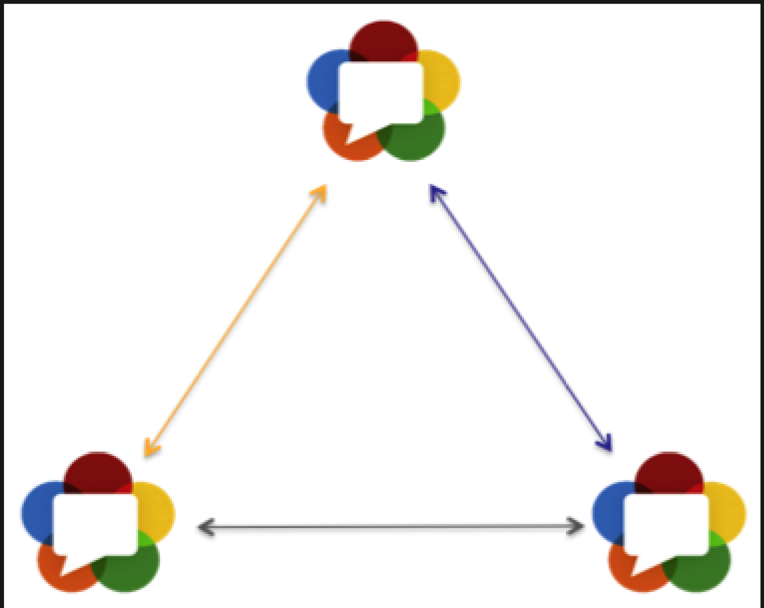


Note: *RTP can be sent over UDP or TCP. Similarly, signalling protocols can be designed to transmit over UDP or TCP.

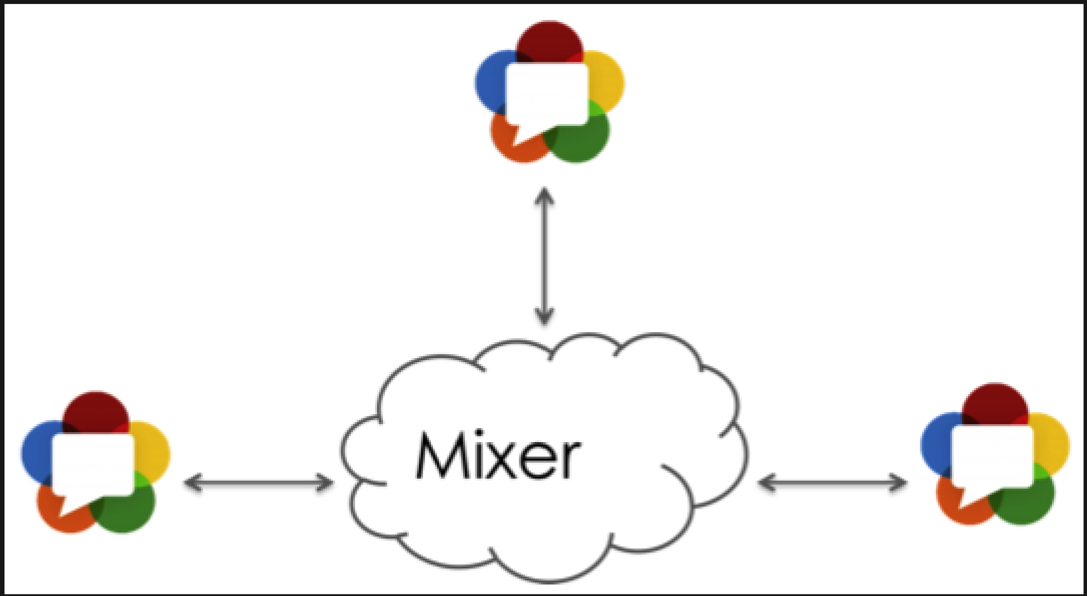
VIDEOCONFERENZA

- Mesh (Peer-To-Peer)
- Mixer (MCU)
- Router (SFU)

PEER-TO-PEER

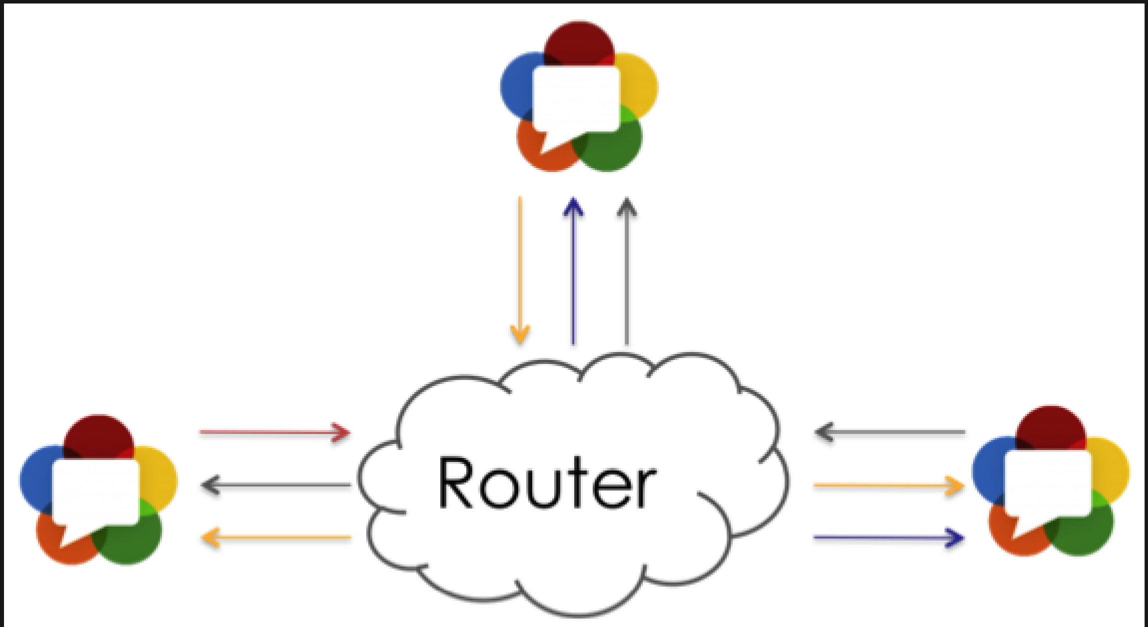


MCU



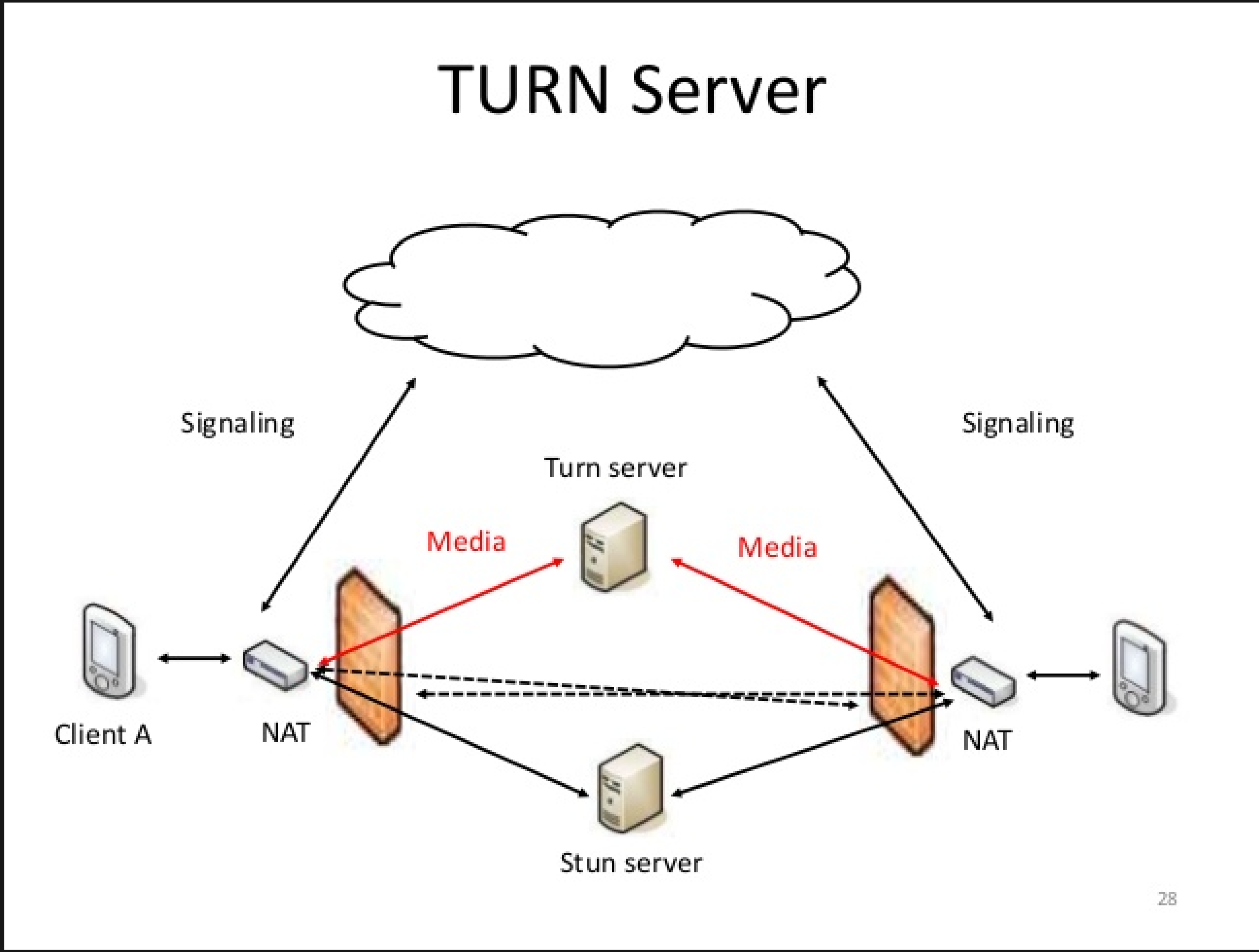
Multipoint Control Unit

SFU

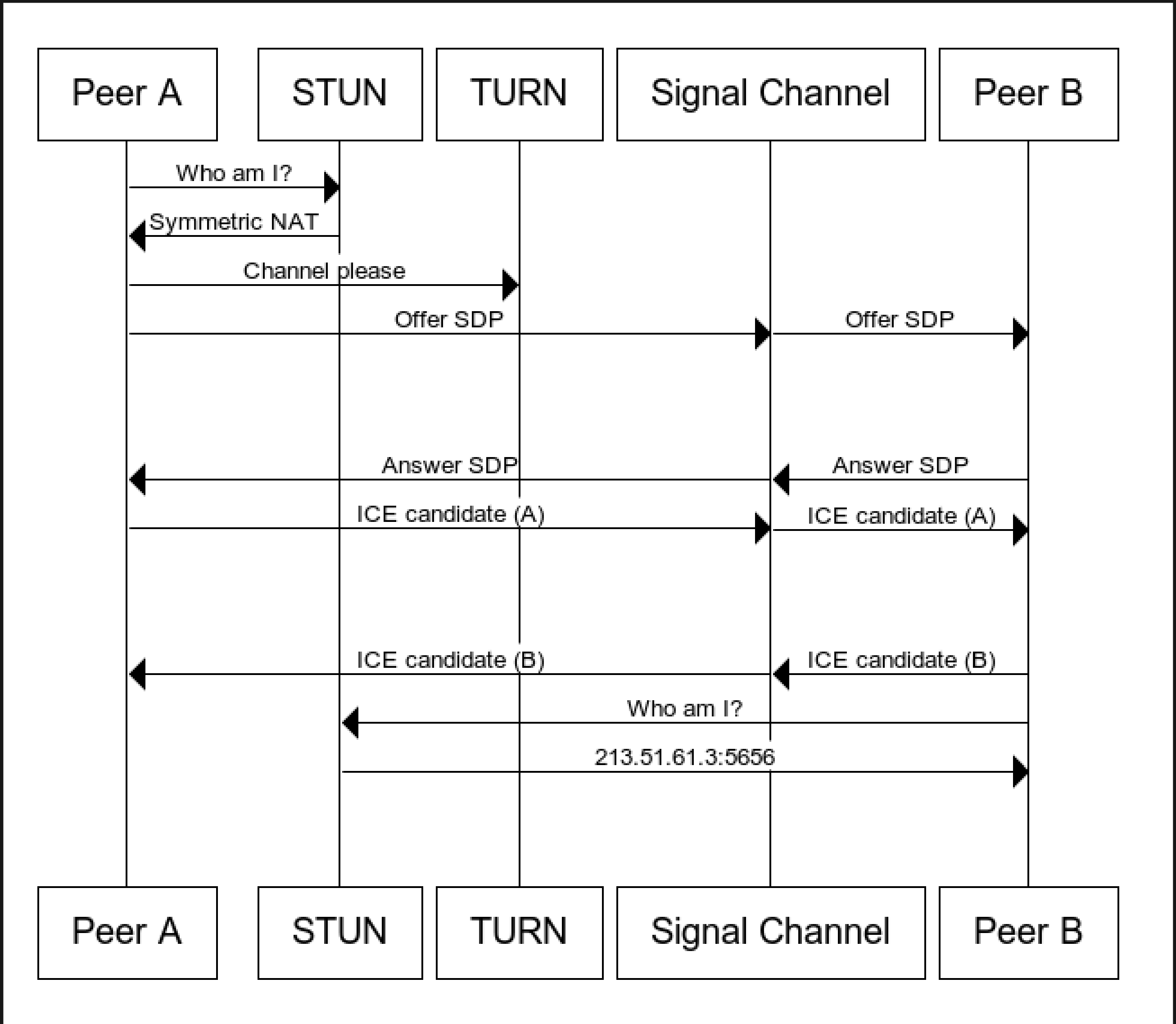


Selective Forwarding Unit

ATTRAVERSARE NAT



ICE


Peer A
STUN
TURN
Signal Channel
Peer B
Who am I?
Symmetric NAT
Channel please
Offer SDP
Offer SDP
Answer SDP
Answer SDP
ICE candidate (A)
ICE candidate (A)
ICE candidate (B)
ICE candidate (B)
Who am I?
213.51.61.3:5656
Peer A
STUN
TURN
Signal Channel
Peer B

DOMANDE?