

Telegram Mobile Protocol in Go

최종석

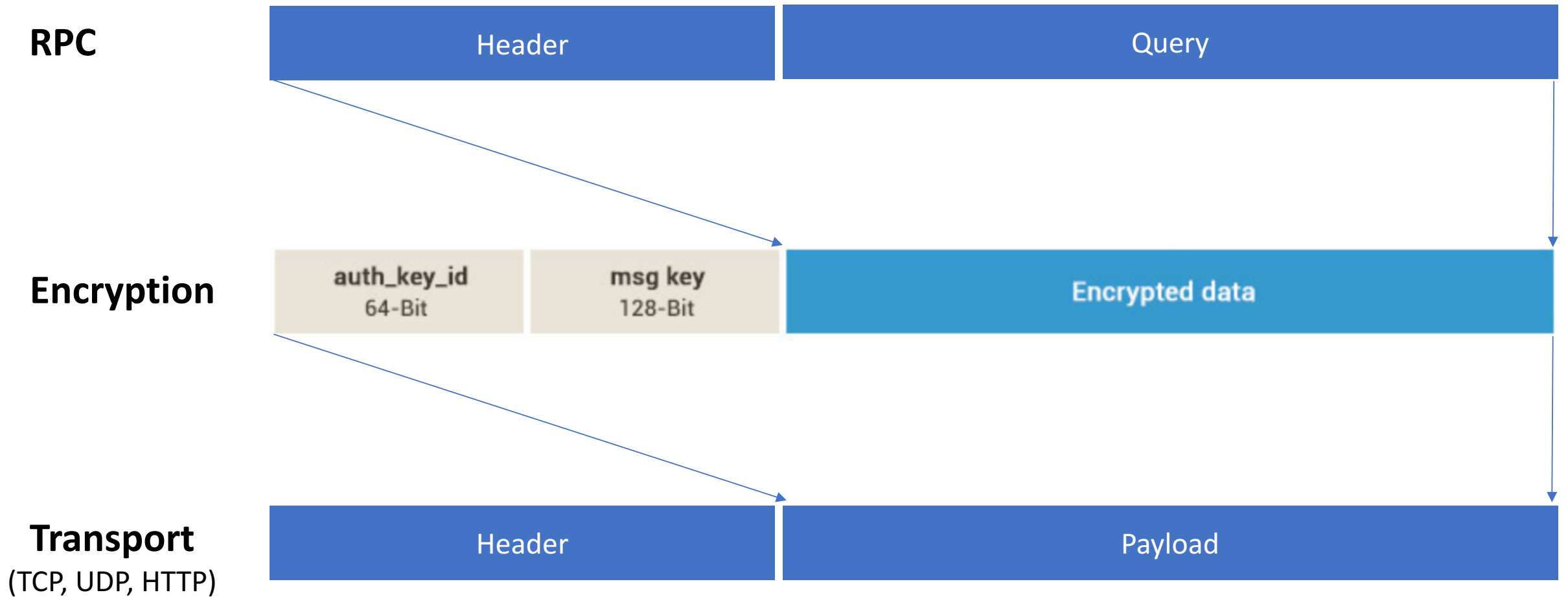
jongseok@6igs.com

Telegram Mobile Protocol을
어디에 쓰는가?

목차

- MTPROTO: Telegram Mobile Protocol
- 텔레그램은 MTPROTO를 어떻게 제공해 왔는가?
- 개발자들은 MTPROTO를 어떻게 구현해 왔는가?
- 나는 MTPROTO를 왜 구현했는가?
 - one 계정 in 멀티머신
 - 모든 RPC의 함수화
 - 커넥션관리
 - 텔레그램 메시지 수신 딜레이 개선

MTPROTO : Telegram Mobile Protocol



MTPROTO : Telegram Mobile Protocol

```
messages.sendMessage#4cde0aab peer:InputPeer message:string random_id:long = messages.SentMessage;
```

RPC



Encryption



Transport
(TCP, UDP, HTTP)



MTPROTO

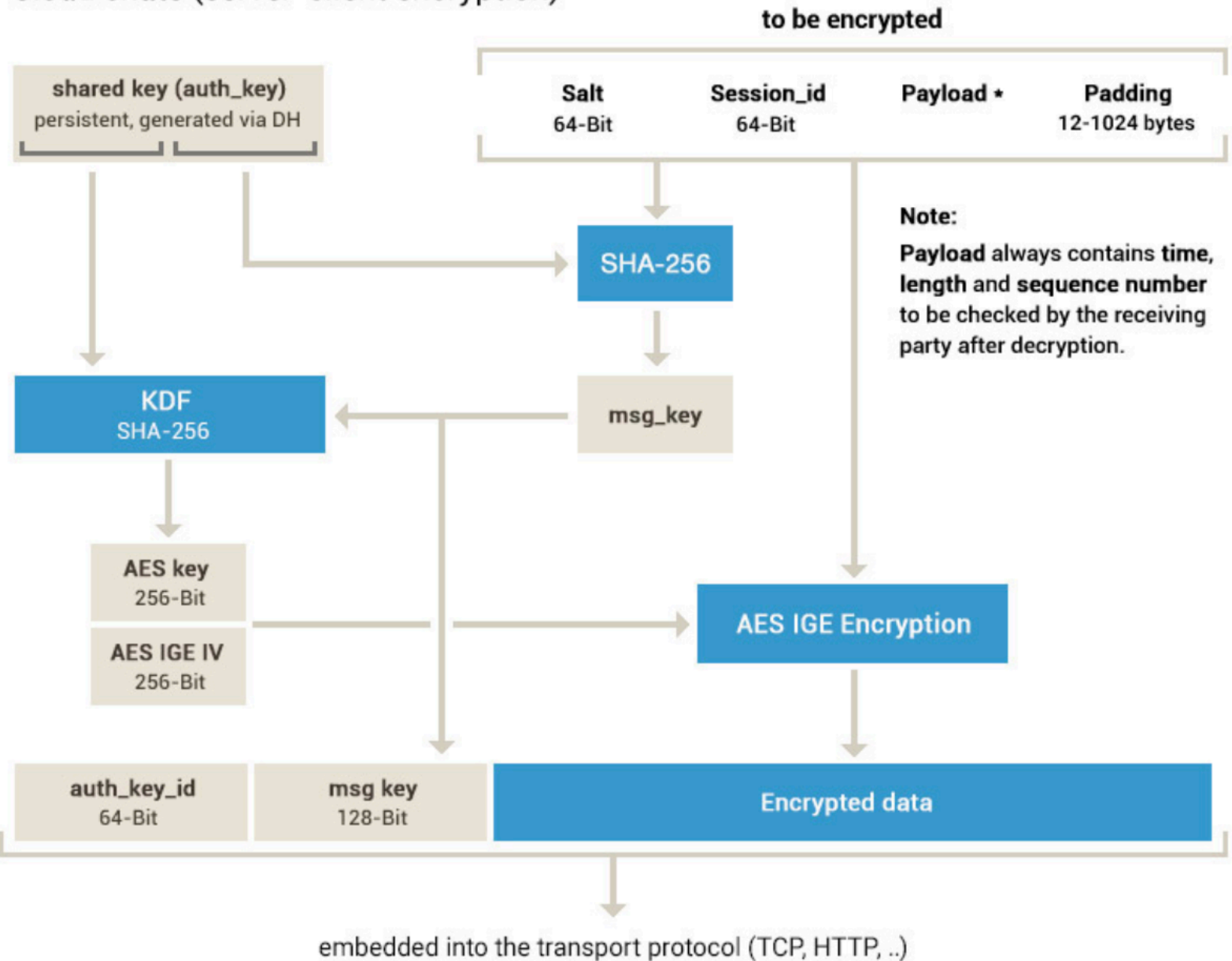
RPC

Encryption

Transport
(TCP, UDP, HTTP)

MTPROTO 2.0, part I

Cloud chats (server-client encryption)



텔레그램은 MTProto를
어떻게 제공해 왔는가?

프로토콜 명세보고 구현하거나
깃헙에 있는거 쓰도록 권고

프로토콜 명세보고 구현하거나
깃헙에 있는거 쓰도록 권고



현재는 RPC over JSON 가능한 공식 C++ 구현체 제공

(repository creation date: 2017.12.31)

<https://github.com/tdlib/td>

원하는 언어에서 Native Interface를 통해 사용하도록 권고

Current TL-schema

Below you will find the current TL-schema. [More details on TL](#) »

See also the [detailed schema in JSON](#) »

See also [TL-Schema for end-to-end encrypted messages](#) »

Layer 23 ▾

```
boolFalse#bc799737 = Bool;  
boolTrue#997275b5 = Bool;  
  
true#3fedd339 = True;  
  
vector#1cb5c415 {t:Type} # [ t ] = Vector t;  
  
error#c4b9f9bb code:int text:string = Error;
```

```
1328  
1329 langpack.getLangPack#f2f2330a lang_pack:  
1330 langpack.getStrings#efea3803 lang_pack:s  
1331 langpack.getDifference#b2e4d7d from_vers  
1332 langpack.getLanguages#42c6978f lang_pack  
1333  
1334 // LAYER 86
```

개발자들은 MTProto를
어떻게 구현해 왔는가?

Repositories	198
Code	47K
Commits	2K
Issues	1K
Marketplace	
Topics	6
Wikis	15
Users	6

Languages	
Go	24
JavaScript	24
Shell	19
Python	18
Java	14

198 repository results

Sort: Best match ▾

[zerobias/telegram-mtproto](#)

JavaScript

★ 437

Telegram client api (*MTPProto*) library

mtproto telegram telegram-api

MIT license Updated 24 days ago

[alexbers/mtprotoproxy](#)

Python

★ 278

Async *mtproto* proxy for Telegram

MIT license Updated 2 days ago

[LonamiWebs/Telethon](#)

Python

★ 2.1k

Pure Python 3 *MTPProto* API Telegram client library

python library telegram python-library

나는 왜 MTPProto를
구현했는가?

요구사항들

- one 계정 in 멀티머신
- 모든 RPC의 함수화
- 커넥션관리
- 텔레그램 메시지 수신 딜레이 개선

요구사항들

- **one 계정 in 멀티머신**
- 모든 RPC의 함수화
- 커넥션관리
- 텔레그램 메시지 수신 딜레이 개선

one 계정 in 멀티머신

- 메시지 수신은 한 계정당 최근 10개 세션으로 제한

Thereafter, whenever there is an event that the user needs to be notified of, the server will find a list of the 10 most recent active sessions and send messages to those sessions. Note that a

<https://core.telegram.org/api/updates>

- 한 계정을 여러 머신에서 사용하면 어뷰징으로 차단 당하지 않을까

one 계정 in 멀티머신

- 메시지 수신은 한 계정당 최근 10개 세션으로 제한

Thereafter, whenever there is an event that the user needs to be notified of, the server will find a list of the 10 most recent active sessions and send messages to those sessions. Note that a

<https://core.telegram.org/api/updates>

- 한 계정을 여러 머신에서 사용하면 어뷰징으로 차단 당하지 않을까



Proxy를 만들자

MTPROTO Proxy

Proxy
Client 1

Proxy
Client 2

...

Proxy
Client n

MTPROTO
Proxy



Telegram

MTPROTO Proxy

Proxy
Client 1

Proxy
Client 2

...

Proxy
Client n

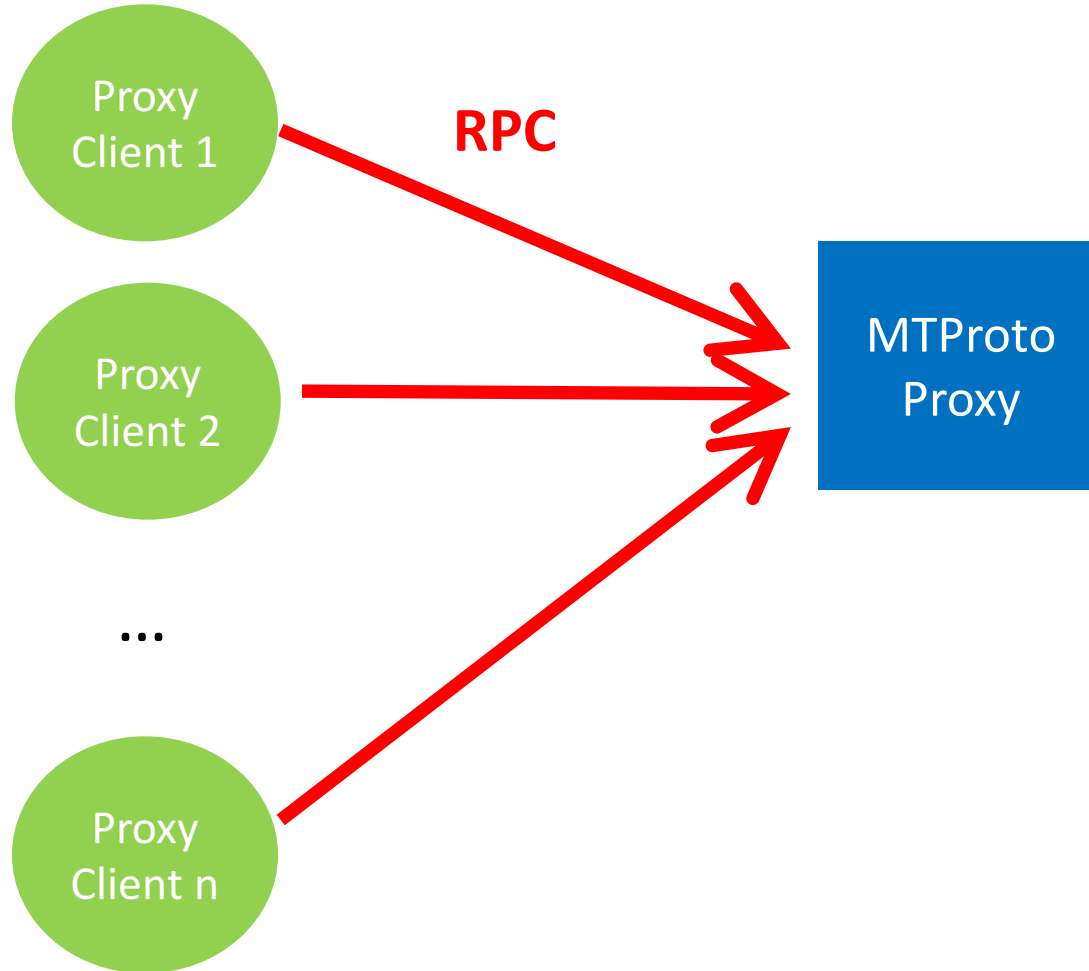
MTPROTO
Proxy

Sign-in

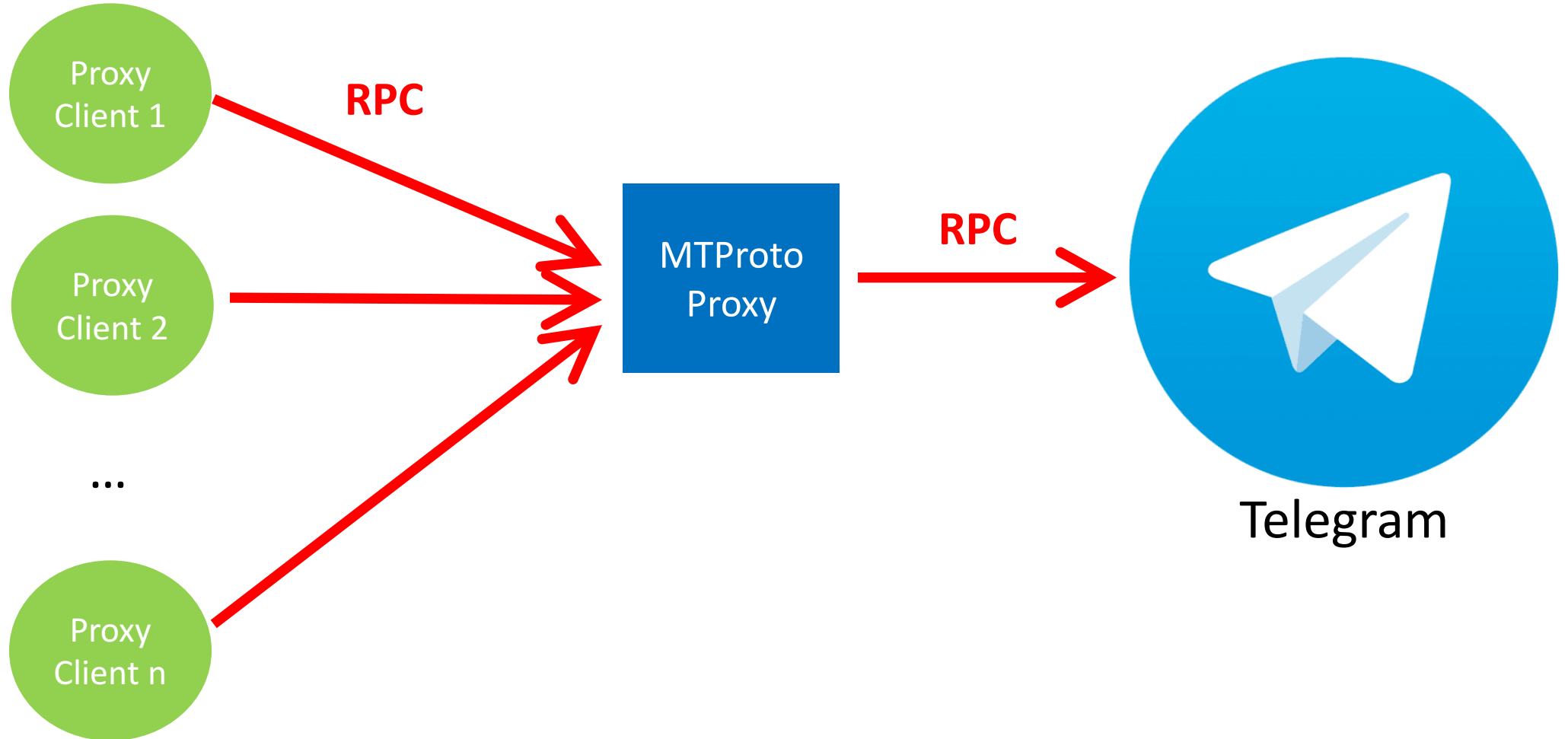


Telegram

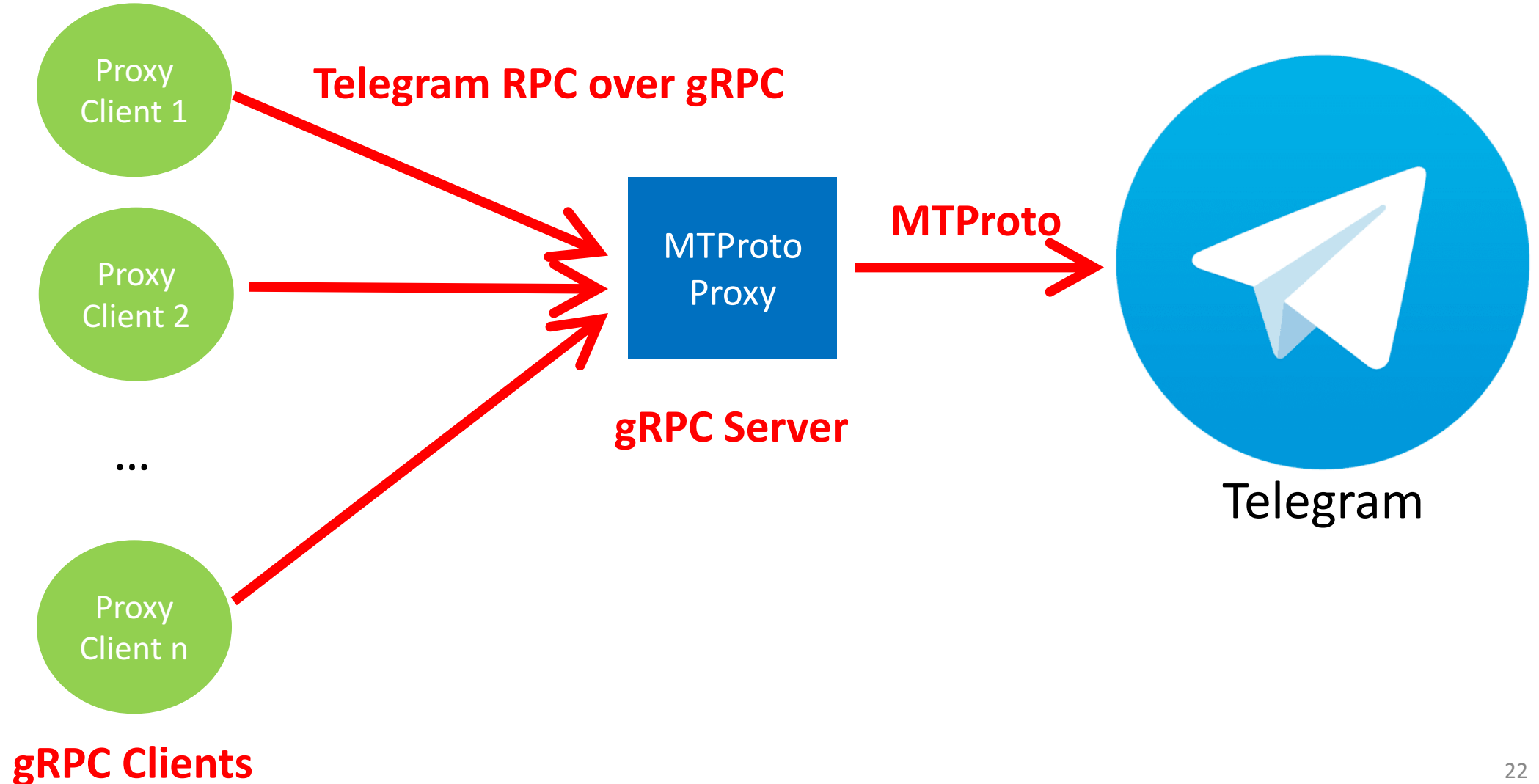
MTPROTO Proxy



MTPROTO Proxy

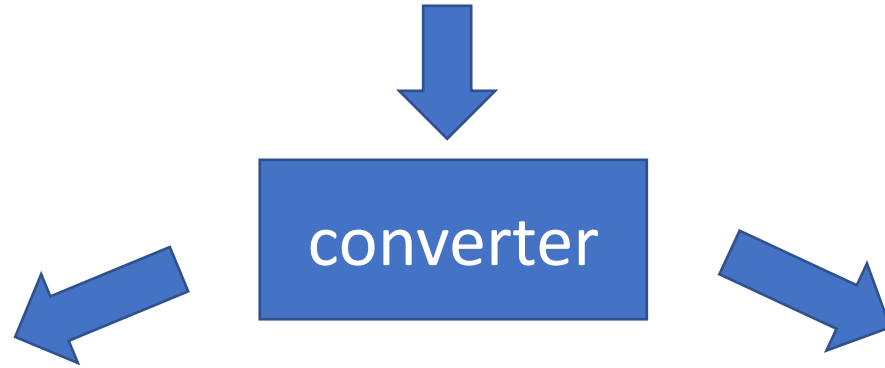


MTPROTO Proxy



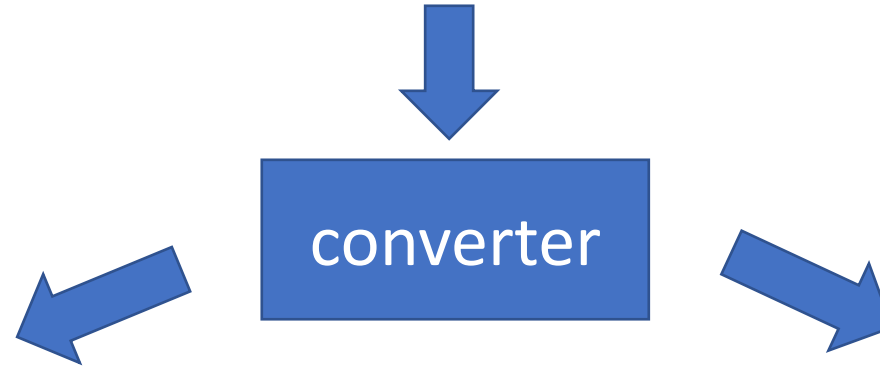
TL-Schema

```
messages.sendMessage#4cde0aab peer:InputPeer message:string random_id:long = messages.SentMessage;
```



TL-Schema

```
messages.sendMessage#4cde0aab peer:InputPeer message:string random_id:long = messages.SentMessage;
```



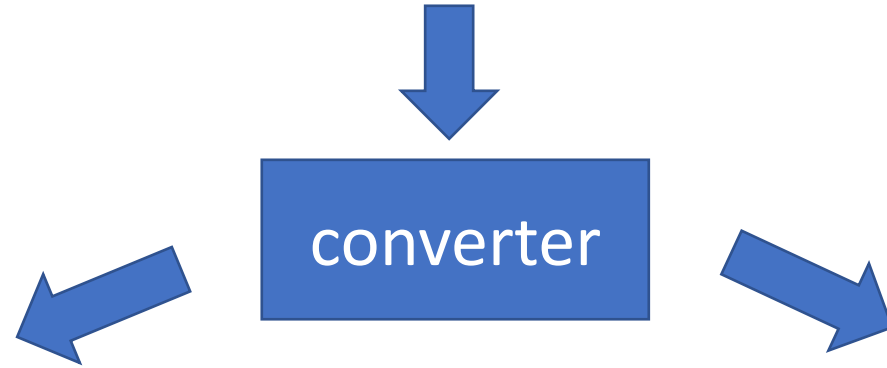
Protocol Buffer codes

```
rpc MessagesSendMessage (ReqMessagesSendMessage) returns (TypeUpdates) {}
```

```
message ReqMessagesSendMessage {  
    int32 Flags = 1;  
    TypeInputPeer Peer = 6;  
    int32 ReplyToMsgId = 7;  
    string Message = 8;  
    int64 RandomId = 9;  
    TypeReplyMarkup ReplyMarkup = 10;  
    repeated TypeMessageEntity Entities = 11;  
}
```


TL-Schema

```
messages.sendMessage#4cde0aab peer:InputPeer message:string random_id:long = messages.SentMessage;
```



Protocol Buffer codes

Encoders/Decoders in Go

```
rpc MessagesSendMessage (ReqMessagesSendMessage) returns (TypeUpdates) {}
```

```
message ReqMessagesSendMessage {  
    int32 Flags = 1;  
    TypeInputPeer Peer = 6;  
    int32 ReplyToMsgId = 7;  
    string Message = 8;  
    int64 RandomId = 9;  
    TypeReplyMarkup ReplyMarkup = 10;  
    repeated TypeMessageEntity Entities = 11;  
}
```

```
func (e *ReqMessagesSendMessage) encode() []byte {  
    x := NewEncodeBuf(512)  
    x.Uint(crc_messagesSendMessage)  
    x.Int(e.Flags)  
    x.Bytes(e.Peer.encode())  
    x.FlaggedInt(e.Flags, 0, e.ReplyToMsgId)  
    x.String(e.Message)  
    x.Long(e.RandomId)  
    x.FlaggedObject(e.Flags, 2, e.ReplyMarkup)  
    x.FlaggedVector(e.Flags, 3, toTLslice(e.Entities))  
    return x.buf  
}
```

요구사항들

- one 계정 in 멀티머신 ➡ **Proxy**
- 모든 RPC의 함수화
- 커넥션관리
- 텔레그램 메시지 수신 딜레이 개선

요구사항들

- one 계정 in 멀티머신 ➡ Proxy
- **모든 RPC의 함수화**
- 커넥션관리
- 텔레그램 메시지 수신 딜레이 개선

모든 RPC의 함수화

```
messages.sendMessage#4cde0aab peer:InputPeer message:string random_id:long = messages.SentMessage;
```

RPC



Encryption



Transport
(TCP, UDP, HTTP)



```
messages.sendMessage#4cde0aab peer:InputPeer message:string random_id:long = messages.SentMessage;
```

모든 RPC의 함수화

RPC



Encryption



Transport
(TCP, UDP, HTTP)



<https://github.com/sdidyk/mtproto/blob/master/mtproto.go>

Source: <https://core.telegram.org/mtproto>

```
messages.sendMessage#4cde0aab peer:InputPeer message:string random_id:long = messages.SentMessage;
```

모든 RPC의 함수

RPC

Header

Encryption

auth_key_id
64-Bit

Transport

(TCP, UDP, HTTP)

Header

```
func (m *MTPProto) SendMessage(user_id int32, msg string) error {  
    resp := make(chan TL, 1)  
    m.queueSend <- packetToSend{  
        TL_messages_sendMessage{  
            TL_inputPeerContact{user_id},  
            msg,  
            rand.Int63(),  
        },  
        resp,  
    }  
    x := <-resp  
    _, ok := x.(TL_messages_sendMessage)  
    if !ok {  
        return fmt.Errorf("RPC: %#v", x)  
    }  
  
    return nil  
}
```

<https://github.com/sdidyk/mtproto/blob/master/mtproto.go>

Source: <https://core.telegram.org/mtproto>

```
messages.sendMessage#4cde0aab peer:InputPeer message:string random_id:long = messages.SentMessage;
```

기존 구현체는 구조체는
TL-Schema를 파싱하여
자동으로 생성하지만,

RPC 는

1. Wrapping 하는 함수를
일일이 만들어
제공하거나

2. RPC에 해당하는
구조체를 전송하도록 함

```
func (m *MTPROTO) SendMessage(user_id int32, msg string) error {  
    resp := make(chan TL, 1)  
    m.queueSend <- packetToSend{  
        TL_messages_sendMessage{  
            TL_inputPeerContact{user_id},  
            msg,  
            rand.Int63(),  
        },  
        resp,  
    }  
    x := <-resp  
    _, ok := x.(TL_messages_sendMessage)  
    if !ok {  
        return fmt.Errorf("RPC: %#v", x)  
    }  
  
    return nil  
}
```

<https://github.com/sdidyk/mtproto/blob/master/mtproto.go>

Source: <https://core.telegram.org/mtproto>

```
messages.sendMessage#4cde0aab peer:InputPeer message:string random_id:long = messages.SendMessage;
```

기존 구현체는 구조체는
TL-Schema를 파싱하여
자동으로 생성하지만,

```
func (m *MTPProto) SendMessage(user_id int32, msg string) error {  
    resp := make(chan TL, 1)  
    m.queueSend <- packetToSend{  
        TL_messages_sendMessage{  
            TL_inputPeerContact{user_id},  
            msg,  

```

모든 RPC 함수 구현체도 자동으로 생성하자

제공하거나

2. RPC에 해당하는
구조체를 전송하도록 함

```
_, ok := x.(TL_messages_sendMessage)  
if !ok {  
    return fmt.Errorf("RPC: %#v", x)  
}  
  
return nil  
}
```


TL-Schema

```
messages.sendMessage#4cde0aab peer:InputPeer message:string random_id:long = messages.SentMessage;
```



converter



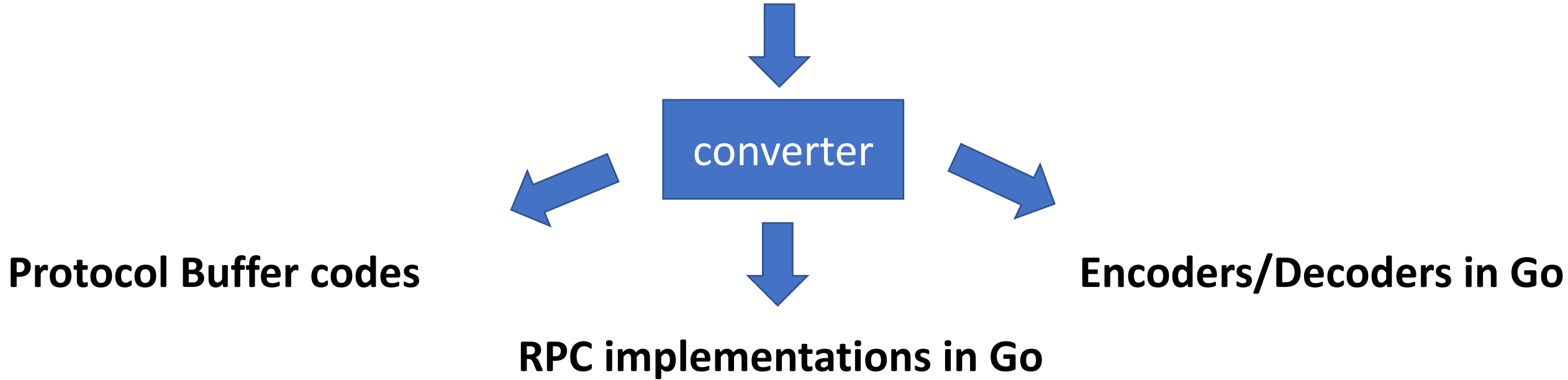
Protocol Buffer codes



Encoders/Decoders in Go

TL-Schema

```
messages.sendMessage#4cde0aab peer:InputPeer message:string random_id:long = messages.SentMessage;
```



rpc MessagesSendMessage (ReqMessagesSendMessage) *returns* (TypeUpdates) {}

```
func (caller RPCCaller) MessagesSendMessage(ctx context.Context, req *ReqMessagesSendMessage) (*TypeUpdates, error) {
    resp, err := caller.RPC.InvokeBlocked(req)
    if err != nil {
        return nil, err
    }
    switch x := resp.(type) {
    case *PredUpdatesTooLong:
        return x.ToType().(*TypeUpdates), nil
    case *PredUpdateShortMessage:
        return x.ToType().(*TypeUpdates), nil
    case *PredUpdateShortChatMessage:
        return x.ToType().(*TypeUpdates), nil
    case *PredUpdateShort:
        return x.ToType().(*TypeUpdates), nil
    case *PredUpdatesCombined:
        return x.ToType().(*TypeUpdates), nil
    case *PredUpdates:
        return x.ToType().(*TypeUpdates), nil
    case *PredUpdateShortSentMessage:
        return x.ToType().(*TypeUpdates), nil
    }
    return &TypeUpdates{}, fmt.Errorf("unexpected return: %T", resp)
}
```



<https://github.com/cjongseok/mtproto/blob/master/procs.tl.go>

rpc MessagesSendMessage (ReqMessagesSendMessage) *returns* (TypeUpdates) {}

```
func (caller RPCCaller) MessagesSendMessage(ctx context.Context, req *ReqMessagesSendMessage) (*TypeUpdates, error) {
    resp, err := caller.RPC.InvokeBlocked(req)
    if err != nil {
        return nil, err
    }
    switch x := resp.(type) {
    case *PredUpdatesTooLong:
        return x.ToType().(*TypeUpdates), nil
    case *PredUpdateShortMessage:
        return x.ToType().(*TypeUpdates), nil
    case *PredUpdateShortChatMessage:
        return x.ToType().(*TypeUpdates), nil
    case *PredUpdateShort:
        return x.ToType().(*TypeUpdates), nil
    case *PredUpdatesCombined:
        return x.ToType().(*TypeUpdates), nil
    case *PredUpdates:
        return x.ToType().(*TypeUpdates), nil
    case *PredUpdateShortSentMessage:
        return x.ToType().(*TypeUpdates), nil
    }
    return &TypeUpdates{}, fmt.Errorf("unexpected return: %T", resp)
}
```

```
message TypeUpdates {
    oneof Value {
        PredUpdatesTooLong UpdatesTooLong = 1;
        PredUpdateShortMessage UpdateShortMessage = 2;
        PredUpdateShortChatMessage UpdateShortChatMessage = 3;
        PredUpdateShort UpdateShort = 4;
        PredUpdatesCombined UpdatesCombined = 5;
        PredUpdates Updates = 6;
        PredUpdateShortSentMessage UpdateShortSentMessage = 7;
    }
}
```

요구사항들

- one 계정 in 멀티머신  Proxy
- 모든 RPC의 함수화  **함수 구현체 자동 생성**
- 커넥션관리
- 텔레그램 메시지 수신 딜레이 개선

요구사항들

- one 계정 in 멀티머신 ➡ Proxy
- 모든 RPC의 함수화 ➡ 함수 구현체 자동 생성
- **커넥션관리**
- 텔레그램 메시지 수신 딜레이 개선

MTProto 커넥션은 서버에 의해서 예고없이 종료가능



기존 구현체는 커넥션을 사용자가 알아서 관리해야함

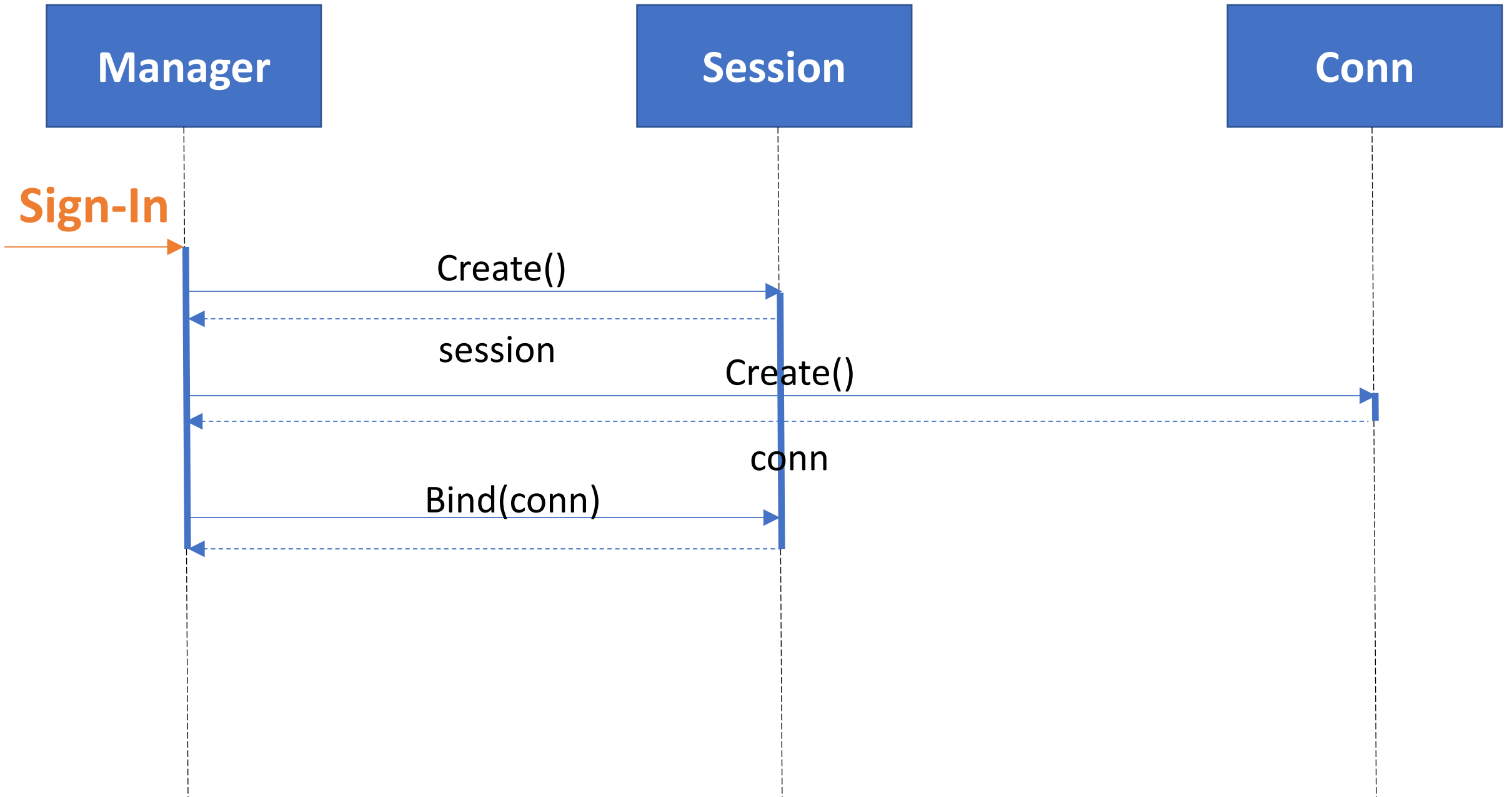
MTProto 커넥션은 서버에 의해서 예고없이 종료가능

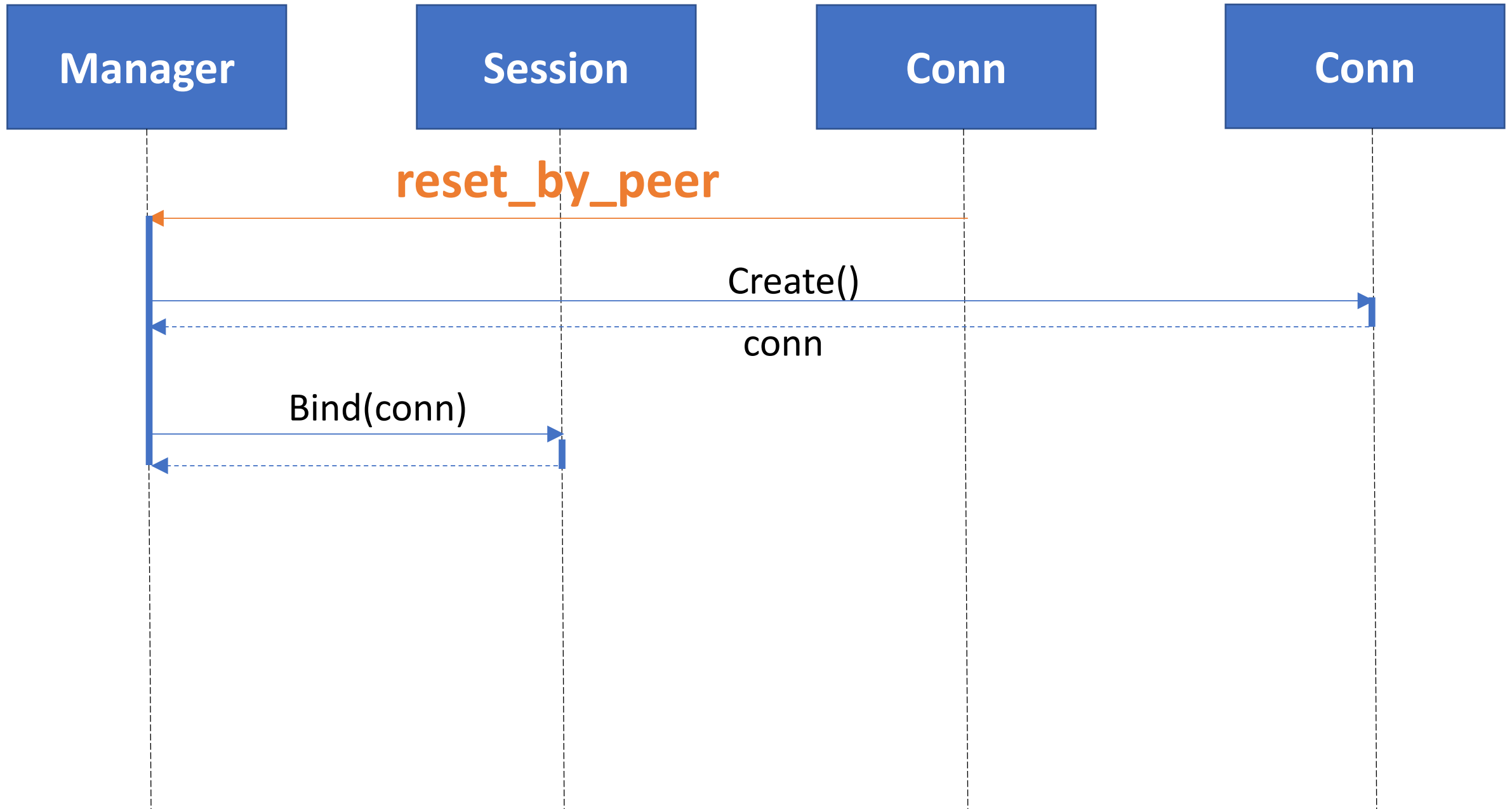


기존 구현체는 커넥션을 사용자가 알아서 관리해야함



**서버에 의해 커넥션 종료시,
이벤트를 발생시켜 자동으로 re-connect!**





요구사항들

- one 계정 in 멀티머신 ➡ Proxy
- 모든 RPC의 함수화 ➡ 함수 구현체 자동 생성
- 커넥션관리 ➡ **Event Driven Connection Manager**
- 텔레그램 메시지 수신 딜레이 개선

요구사항들

- one 계정 in 멀티머신 ➡ Proxy
- 모든 RPC의 함수화 ➡ 함수 구현체 자동 생성
- 커넥션관리 ➡ Event Driven Connection Manager
- 텔레그램 메시지 수신 딜레이 개선

채널에서 메시지를 받는데 짧게는 수초에서 수십초 걸림

채널에서 메시지를 받는데 짧게는 수초에서 수십초 걸림



여러번의 테스트 결과 딜레이가 최근 세션 이용시간과 관계가 있음을 알게됨

채널에서 메시지를 받는데 짧게는 수초에서 수십초 걸림



여러번의 테스트 결과 딜레이가 최근 세션 이용시간과 관계가 있음을 알게됨



Ping RPC를 주기적으로 이용하여 세션 이용시간을 업데이트하자

```
func (conn *Conn) pingRoutine() {
    slog.LogIn(conn, v: "ping: start")
    defer func() {
        conn.isPing = false
        conn.pingWaitGroup.Done()
    }()
    for {
        select {
        case <-conn.pingInterrupter:
            conn.isPing = false
            return
        case <-time.After(conn.appConfig.PingInterval):
            conn.queueSend <- packetToSend{ msg: TL_ping{ ping_id: 0xCADACADA}, resp: nil}
        }
    }
}
```


요구사항들

- one 계정 in 멀티머신 ➡ Proxy
- 모든 RPC의 함수화 ➡ 함수 구현체 자동 생성
- 커넥션관리 ➡ Event Driven Connection Manager
- 텔레그램 메시지 수신 딜레이 개선 ➡ Ping

Demo

감사합니다