



# Интеграция go-сервисов в ICQ

Вольдэмар Дулецкий, команда ICQ

[v.duletskiy@corp.mail.ru](mailto:v.duletskiy@corp.mail.ru)

# Введение

- Много продуктовых задач
- Много уже существующих сервисов со сложившейся инфраструктурой
- Задачи требуют быстрого внедрения
- На C/C++ будет долго

**Чем занимаемся?**

# files.icq.com

хранение файлов

PHP, Golang, MySQL, Tarantool

# go suggest

подсказка ответа стикером

Golang, ML, ipros

14:20

11:55



вчера



13 сент.



Привет 14:20



Сообщение



ПРИВЕТ



## На основе ML

1. предложить заменить вводимый текст стикером
2. ответ стикером на сообщение
3. ответ текстом на сообщение

# Golang, Gitlab, RPM, Puppet

дефолтный стек [mail.ru](https://mail.ru)



# ipproto?

12 байт должно хватить каждому

# Структура пакета

## **ЗАГОЛОВОК**

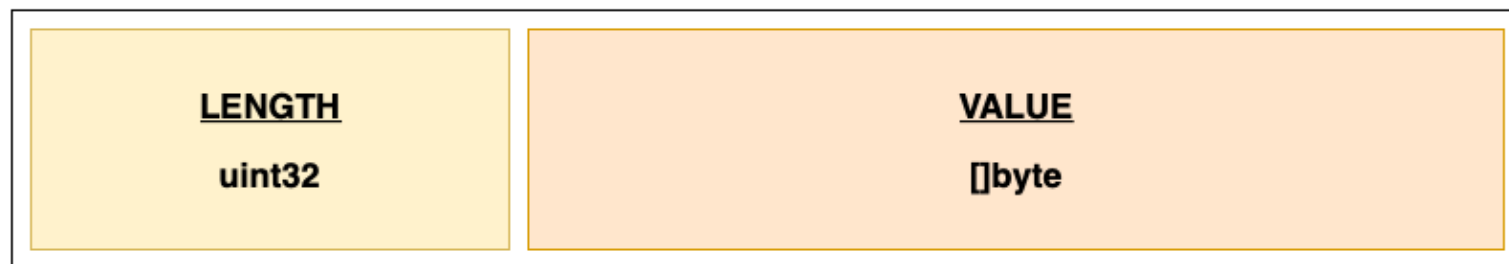
метод (uint32)	длина (uint32)	ID (uint32)
----------------	----------------	-------------

## **ТЕЛО**

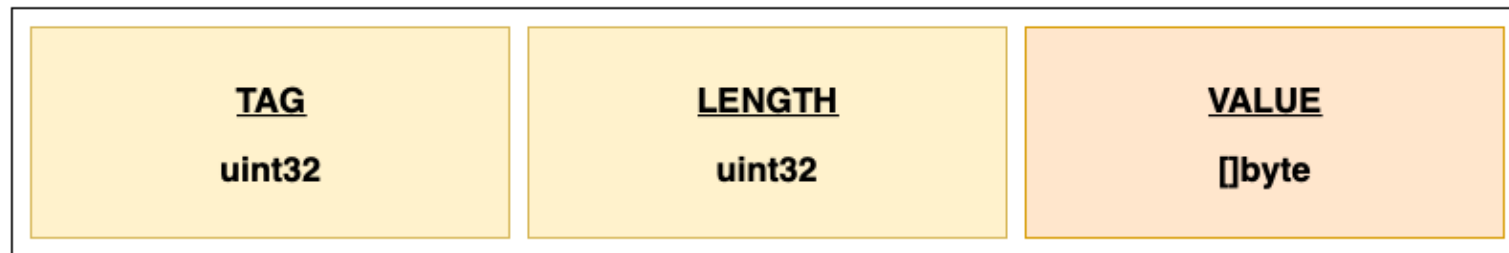
бинарные данные
-----------------

# Содержимое пакета

## SL



## TLV



# TLV

очень полезная структура

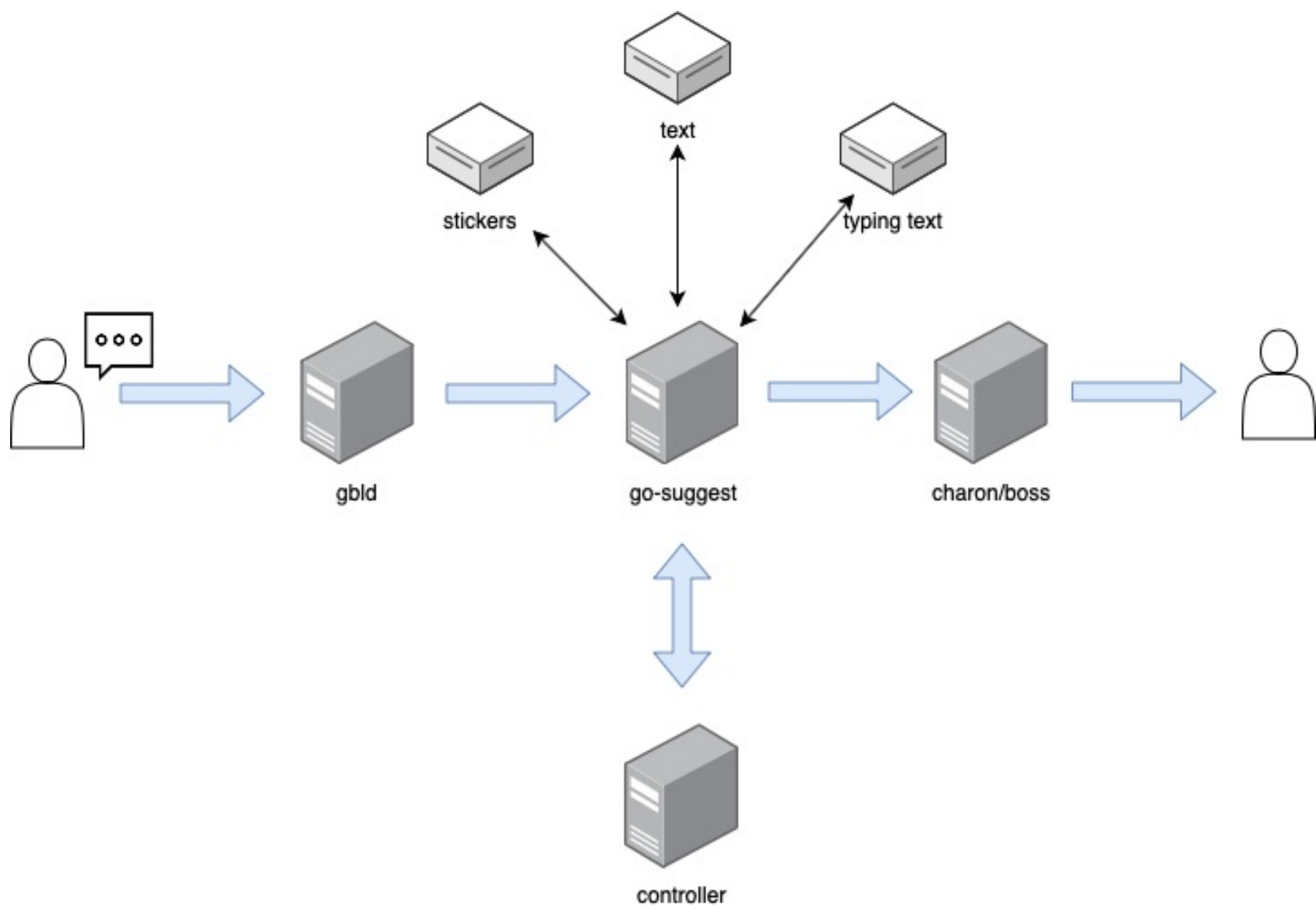
## Пример кода

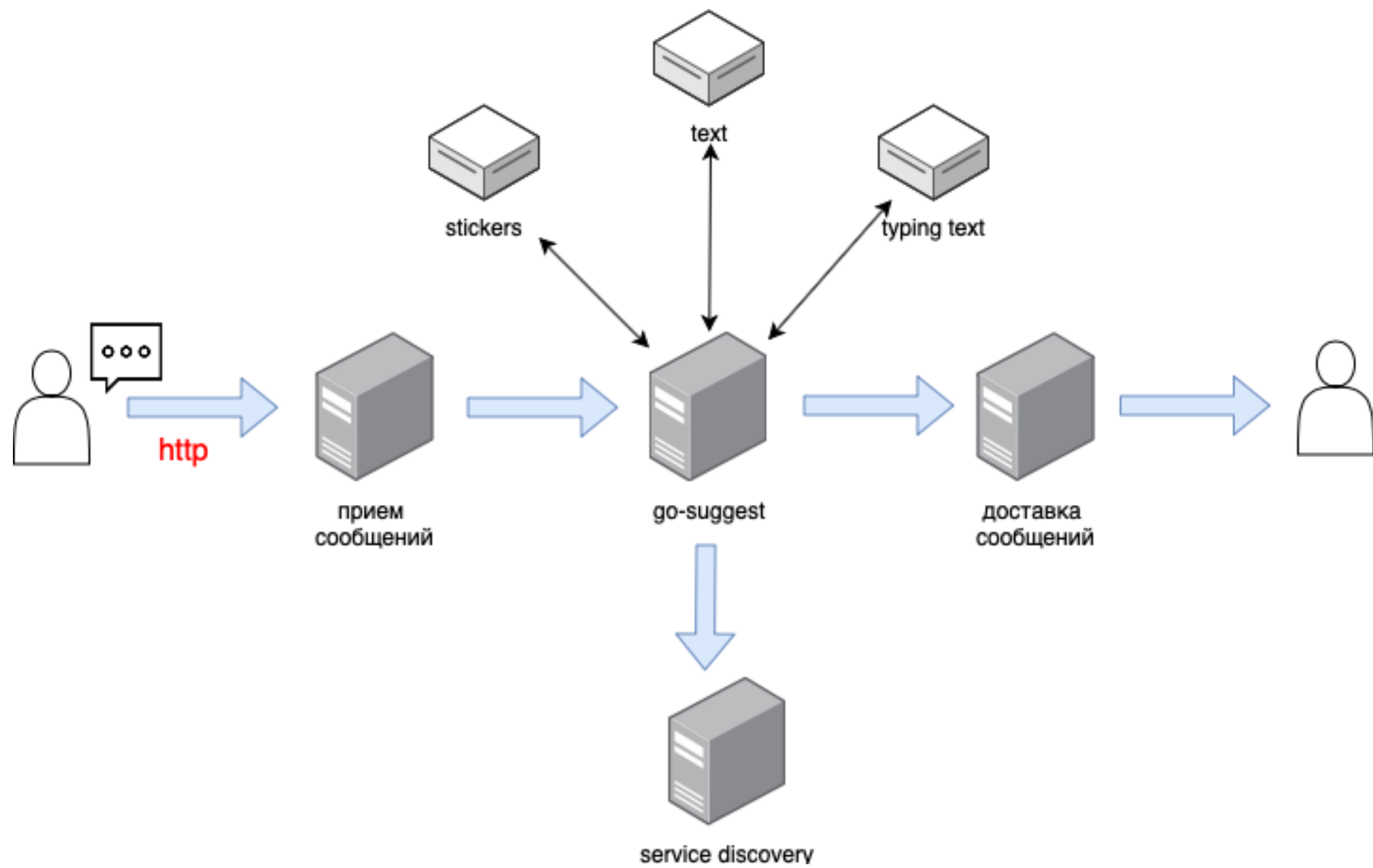
```
func HandshakePayload(svc string, host string, cfg string) []byte {  
    return Concat(  
        SLPack(  
            Concat(  
                SLPack([]byte(svc)),  
                SLPack([]byte(host)),  
                SLPack([]byte(cfg)),  
            ),  
            PackUint32(CNFlags))  
    }
```

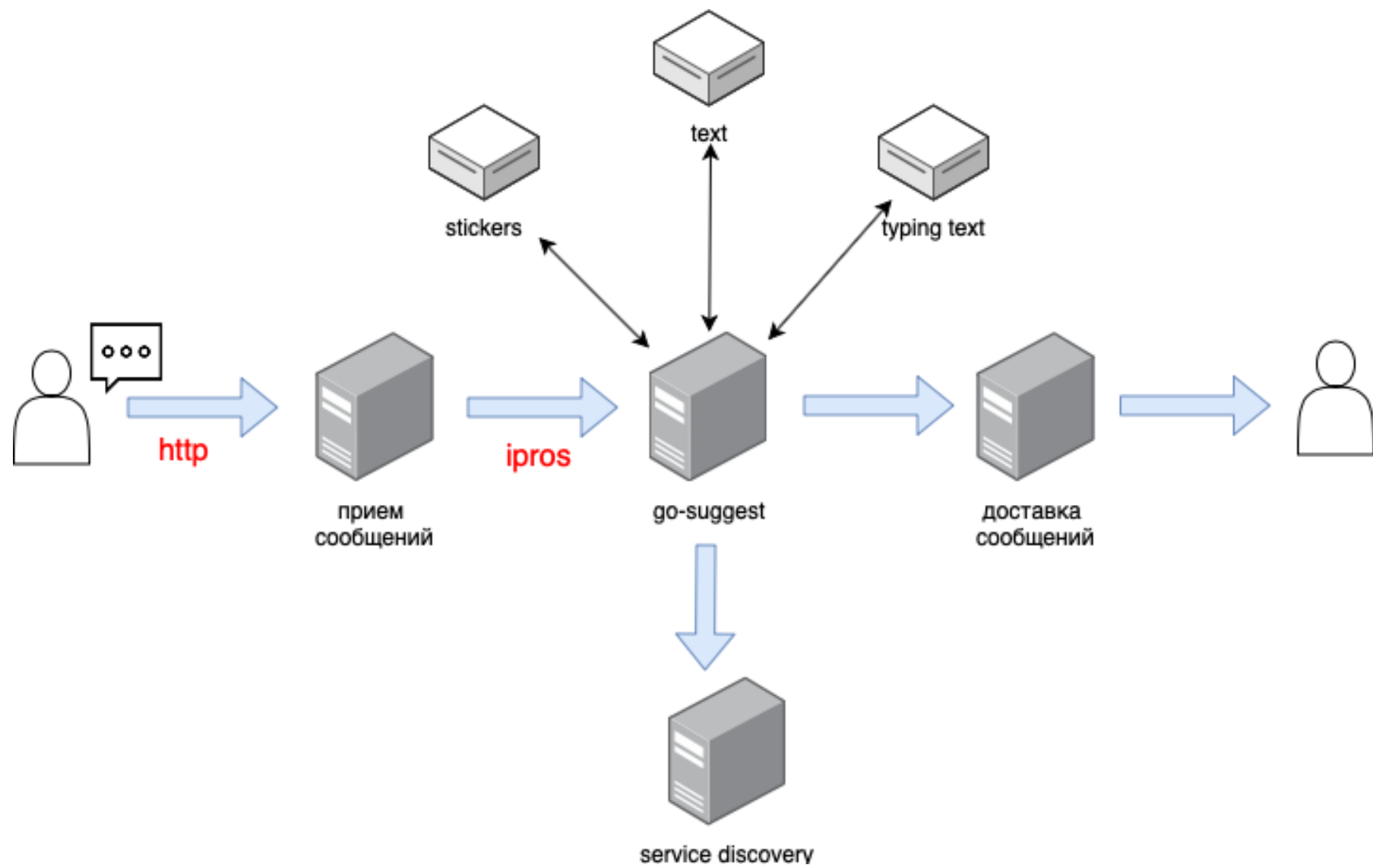
# ipros?

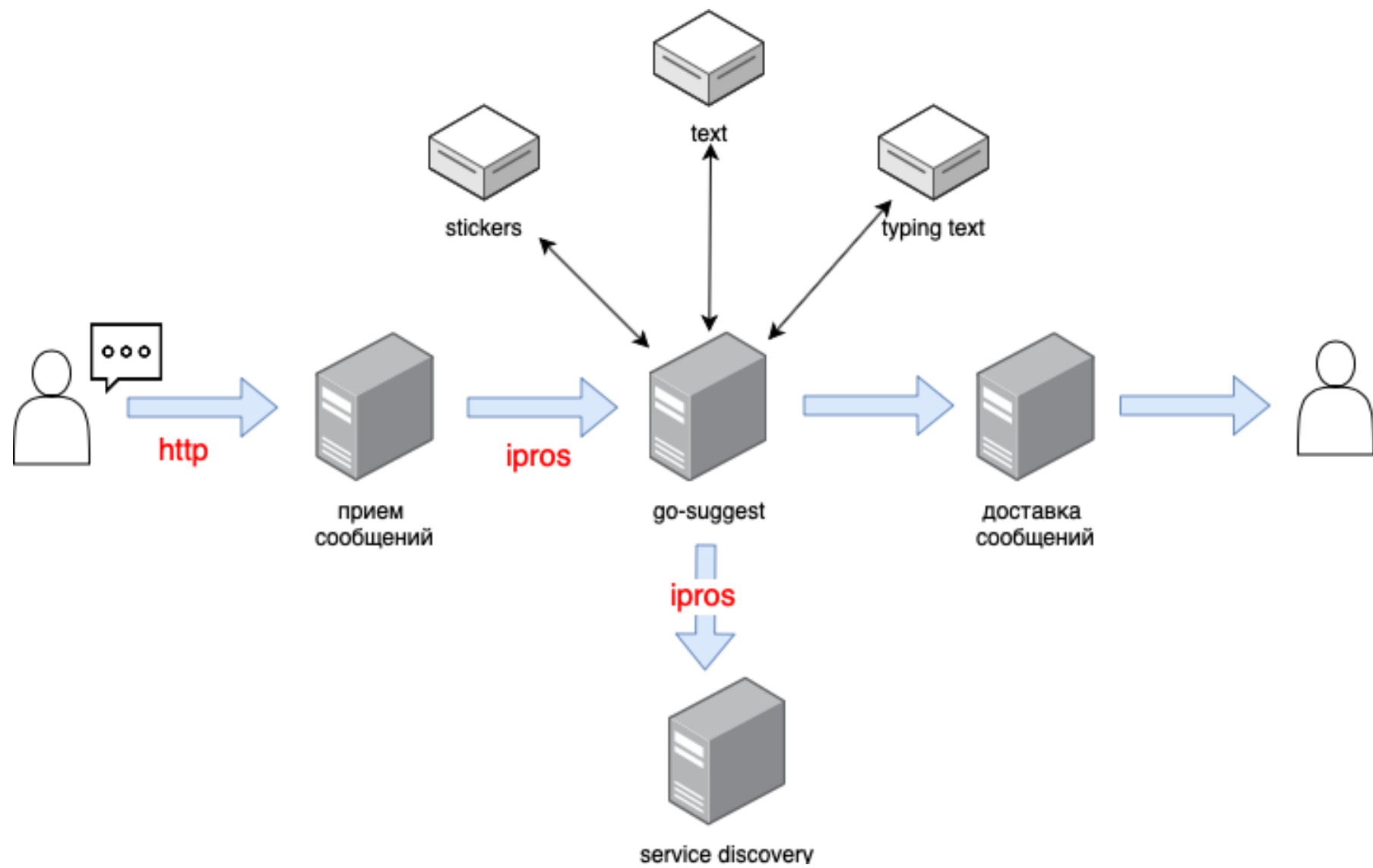
iproto с шардированием

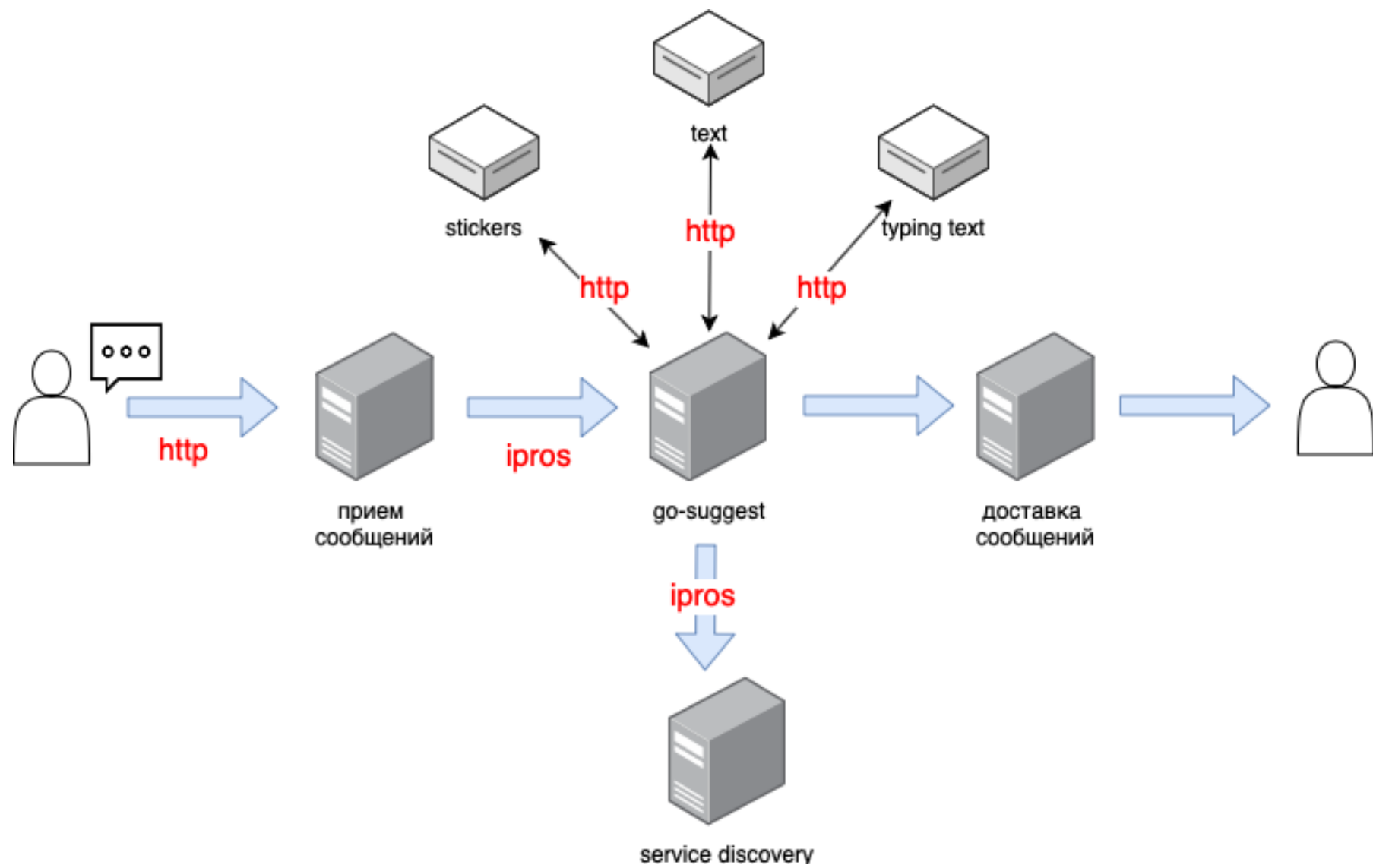


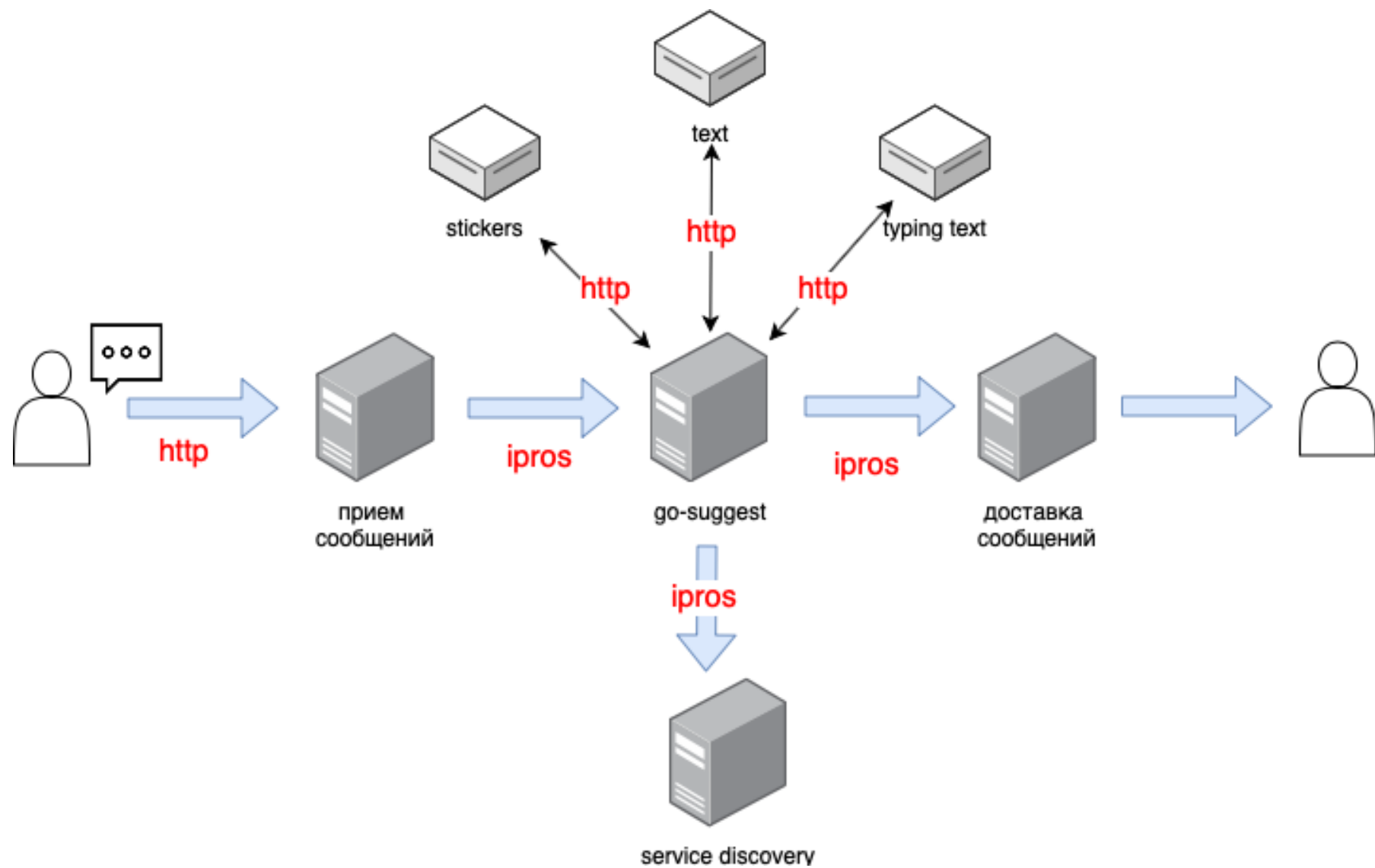


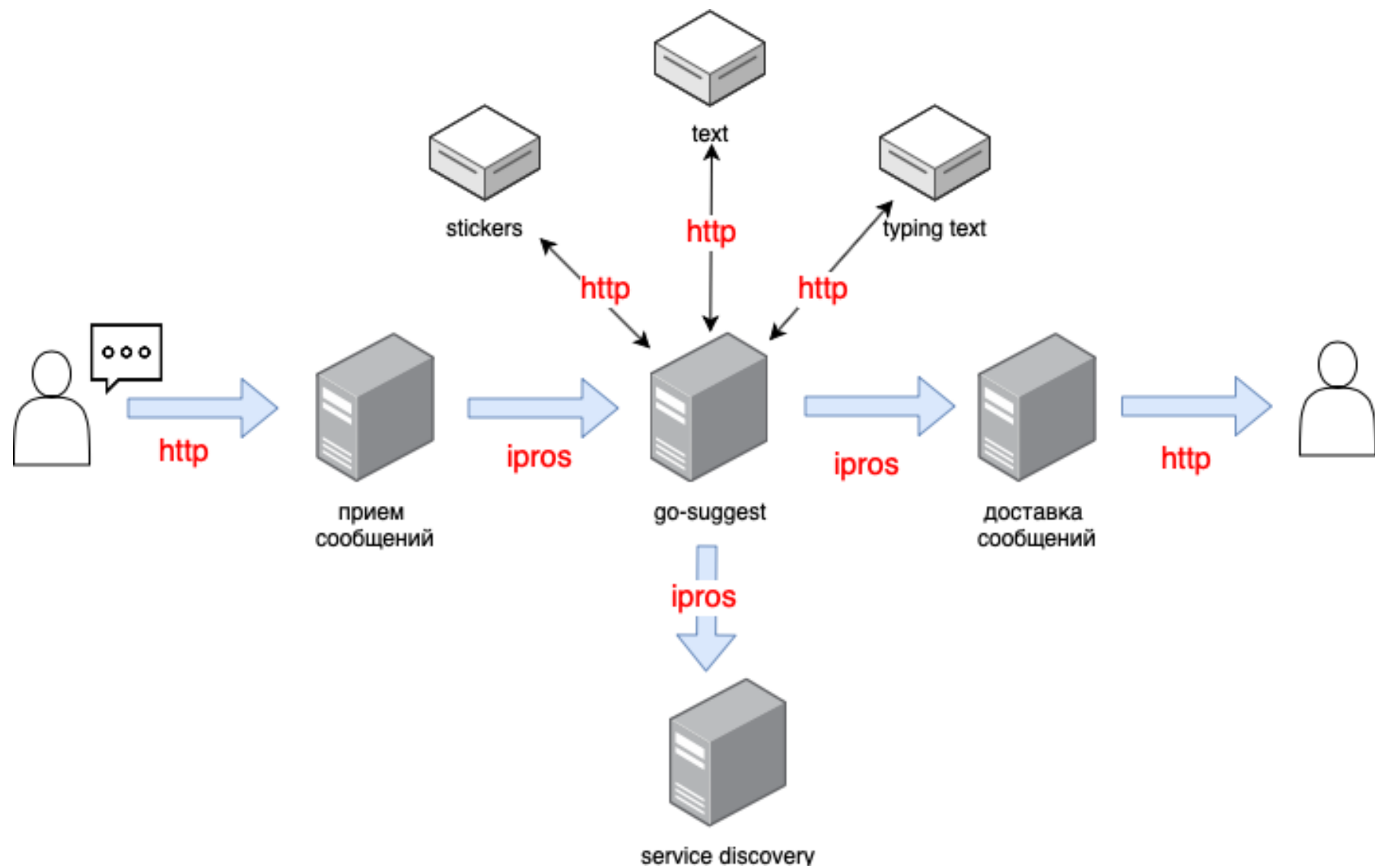












# JSON запрос

```
POST /suggest HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Go-http-client/1.1
Content-Length: 178
Content-Type: application/json
Accept-Encoding: gzip
```

```
{"msgs":
  [
    {"sender":"7105550","receiver":"7105551","TS":1564997991,"text":"тест"},
    {"sender":"7105551","receiver":"7105550","TS":1564997991,"text":"тест"}
  ]
}
```

178 байт



# gRPC запрос

```
message Request {  
    repeated Msg message = 1;  
}  
  
message Msg {  
    string sender      = 1;  
    string receiver    = 2;  
    uint32 TS          = 3;  
    string text         = 4;  
}
```

162 байта

# Ipros запрос

01	00	00	00	7b	00	00	00	42	04	cb	9a	01	00	00	00		....{...B.....
07	00	00	00	37	37	30	31	35	31	34	01	00	00	00	64		....7701514....d
00	00	00	02	00	00	00	5c	00	00	00	03	00	00	00	26		.....\.....&
00	00	00	07	00	00	00	37	31	30	35	35	35	30	07	00		.....7105550..
00	00	37	31	30	35	35	35	31	3d	24	8a	5d	08	00	00		..7105551=\$. ]...
00	d1	82	d0	b5	d1	81	d1	82	03	00	00	00	26	00	00		.....&..
00	07	00	00	00	37	31	30	35	35	35	31	07	00	00	00		.....7105551....
37	31	30	35	35	35	30	3d	24	8a	5d	08	00	00	00	d1		7105550=\$. ].....
82	d0	b5	d1	81	d1	82											.....

123 байта

# ipros соединение

1. соединение с контроллером (service discovery)
2. получение списка хостов, шардированных по ключу
3. выбор хоста
4. handshake
5. обмен данными

# Почему не gRPC?

1. во многих крупных компаниях есть свой RPC
2. ipros экономит трафик
3. исторически так сложилось

# Боль

1. делой pytorch моделей, запакованных в RPM
2. структура ipros сообщений жестко не задается на уровне протокола

# Как избавились от боли?

1. ручная сборка nmslib
2. написана библиотека для работы с бинарными данными
3. ipros: с нуля написан клиент и сервер

# // TODO

```
type Message struct {  
    string Sender      `ipproto:"sl"`  
    string Receiver    `ipproto:"sl"`  
    uint32 TS          `ipproto:". "`  
    string Text        `ipproto:"sl"`  
}  
  
type Request struct {  
    uint32 MsgType     `ipproto:". "`  
    string Sender      `ipproto:"sl"`  
    []Message Messages `ipproto:"sl"`  
}  
...  
  
r := &Request{}  
ipros.Unmarshal(data, r)
```

\*сделать работу с бинарными данными еще проще

## Итог

Увеличение скорости разработки сервисов относительно времени, затрачиваемому на разработку аналогичных сервисов на C/C++.



**Спасибо!**