

# 即时通讯架构

陌陌/李志威/CTO

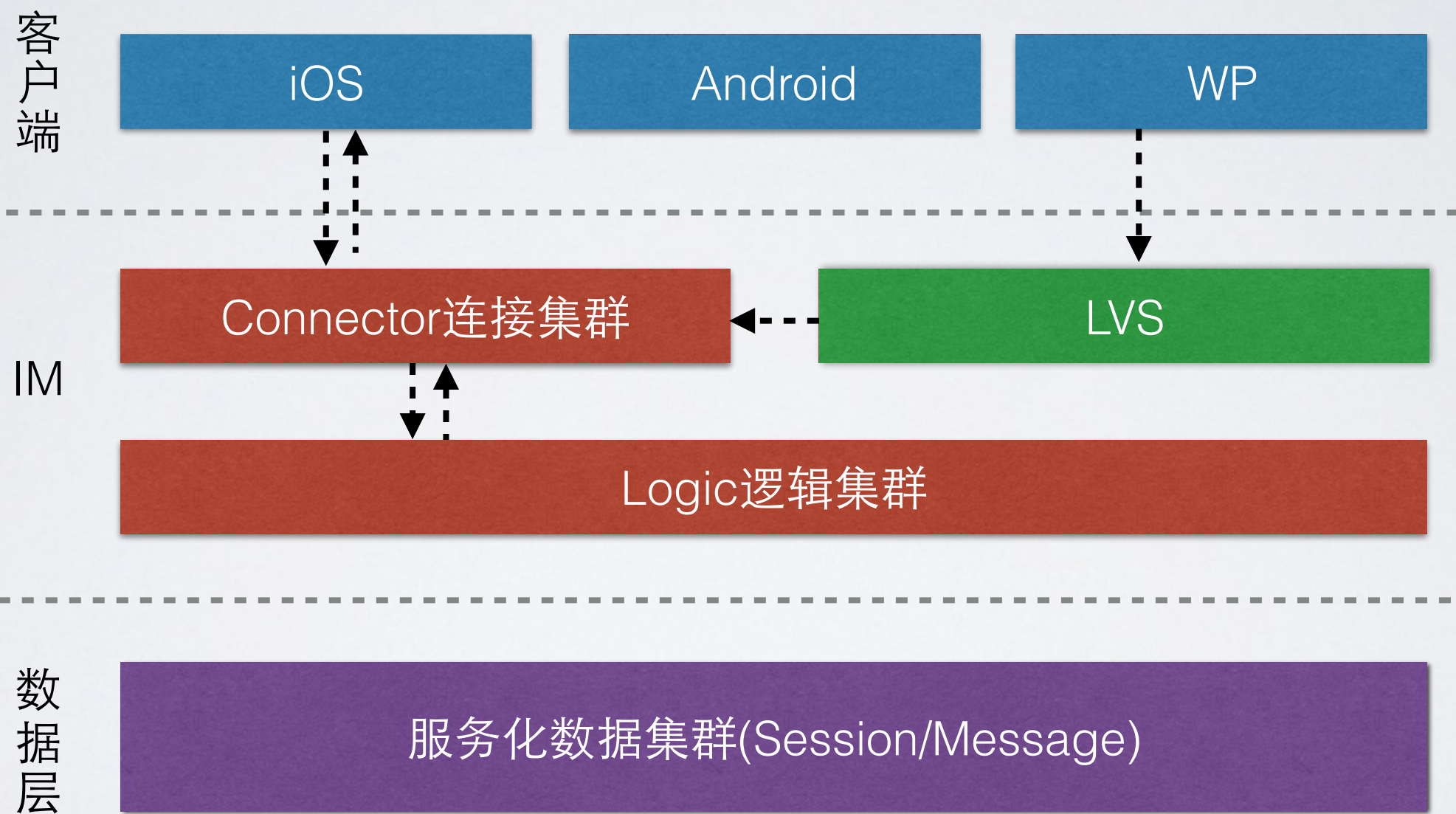


“社交，是人们运用一定的方式(工具)传递信息，  
交流思想，以达到某种目的的社会活动。”

# 常见于

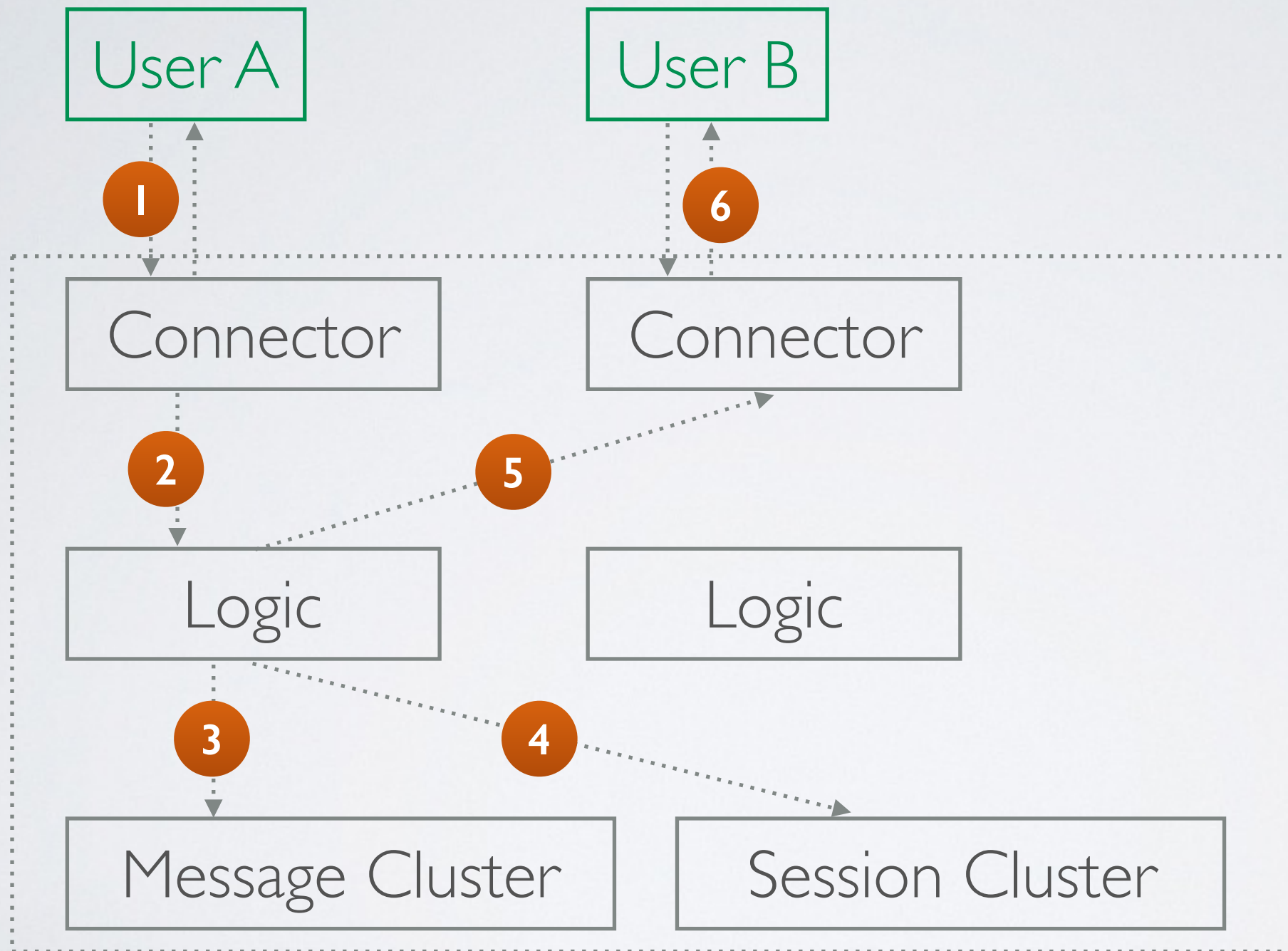
- 即时通讯，私信应用
- 游戏服务器
- 长连接信息推送

# 通讯服务器组成



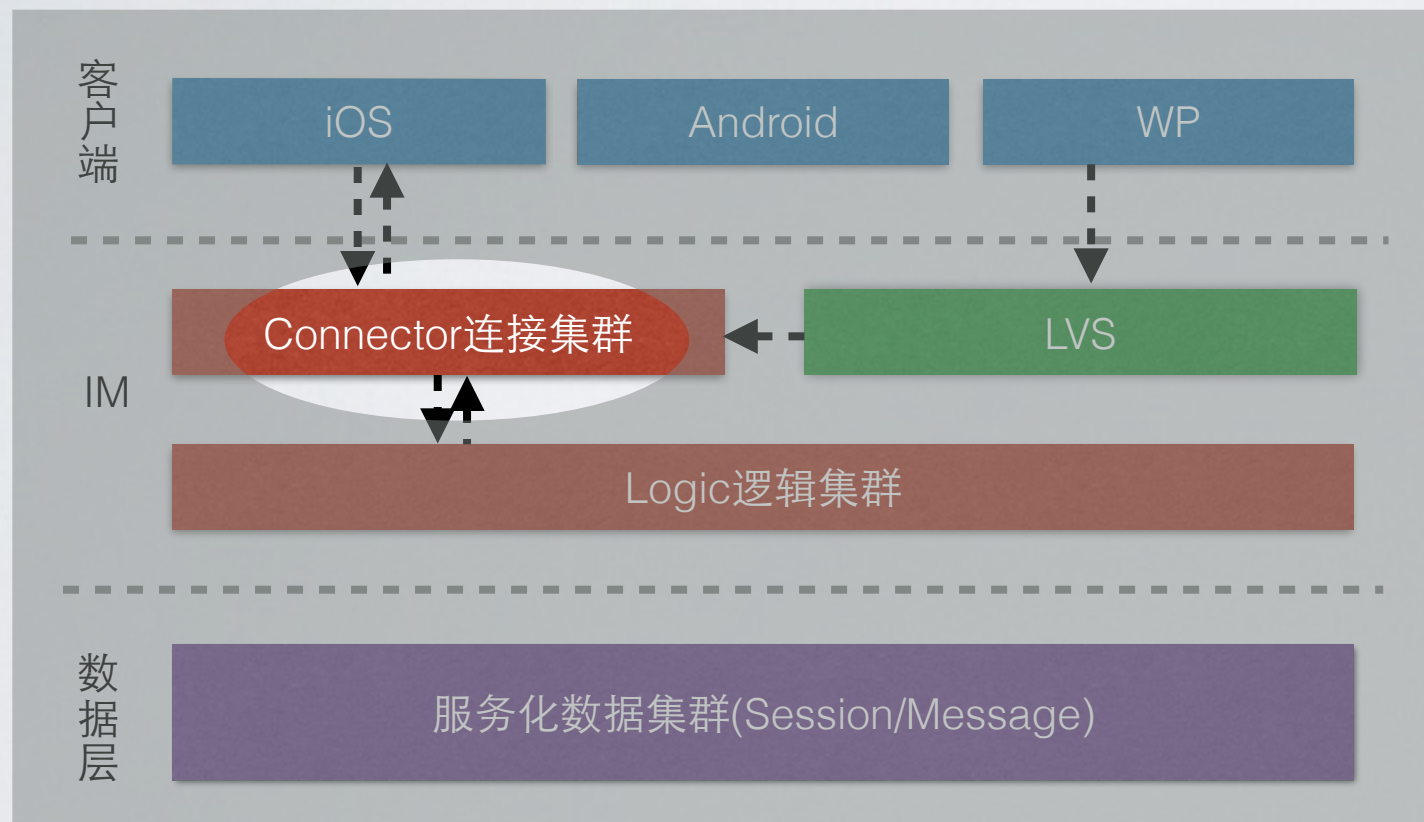


# 消息中转



“对安全性、高可用、扩展性的要求越高，架构的变化越大。”

# 连接层



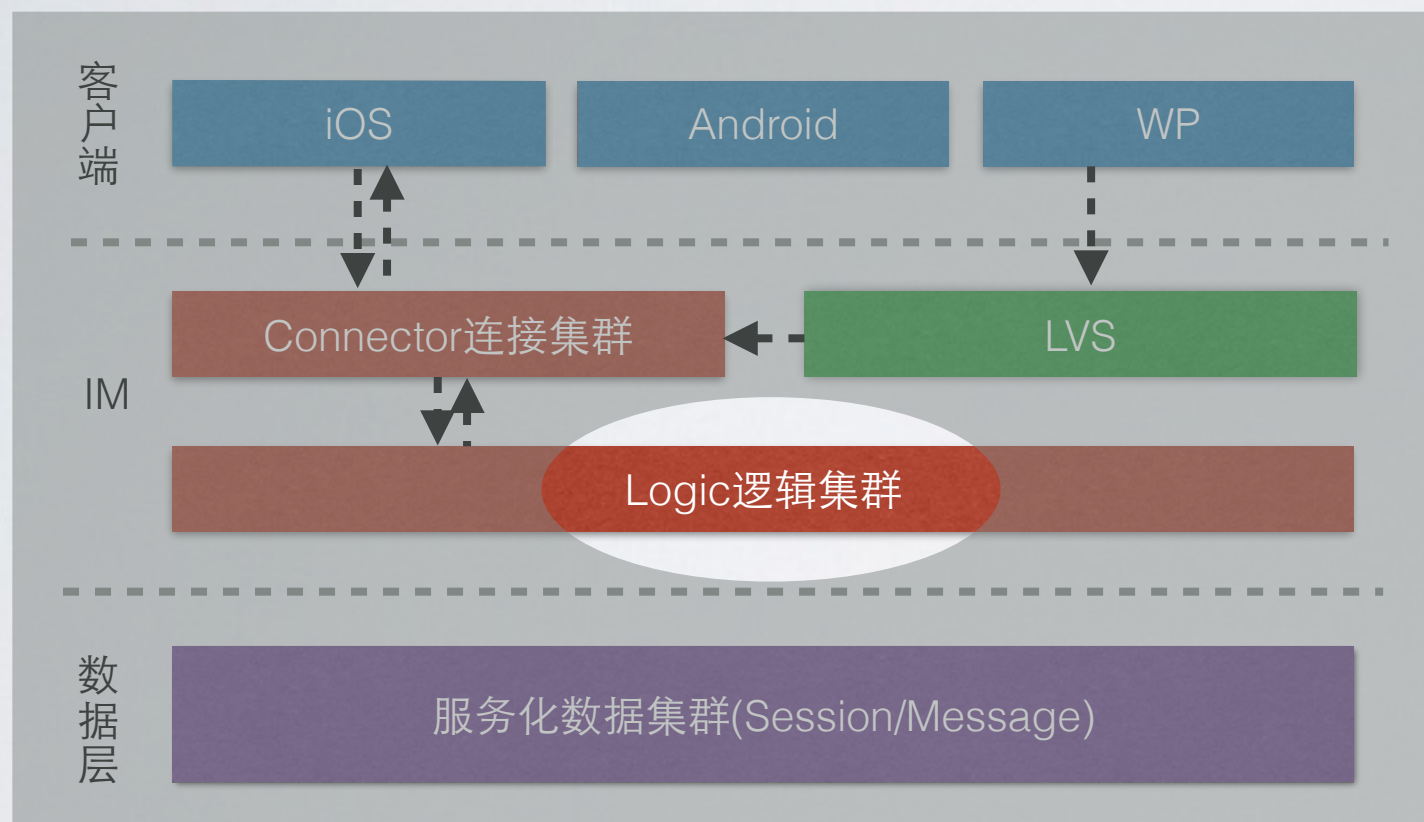
1. 连接层的作用
2. “允许随时重启更新/  
只允许晚上重启/不允许重启断线”
3. 总的来说简单/异步

# 陌陌连接层

- 总连接数 1200万+
- 单台服务器压测70万连接，一般使用50%(主流配置)

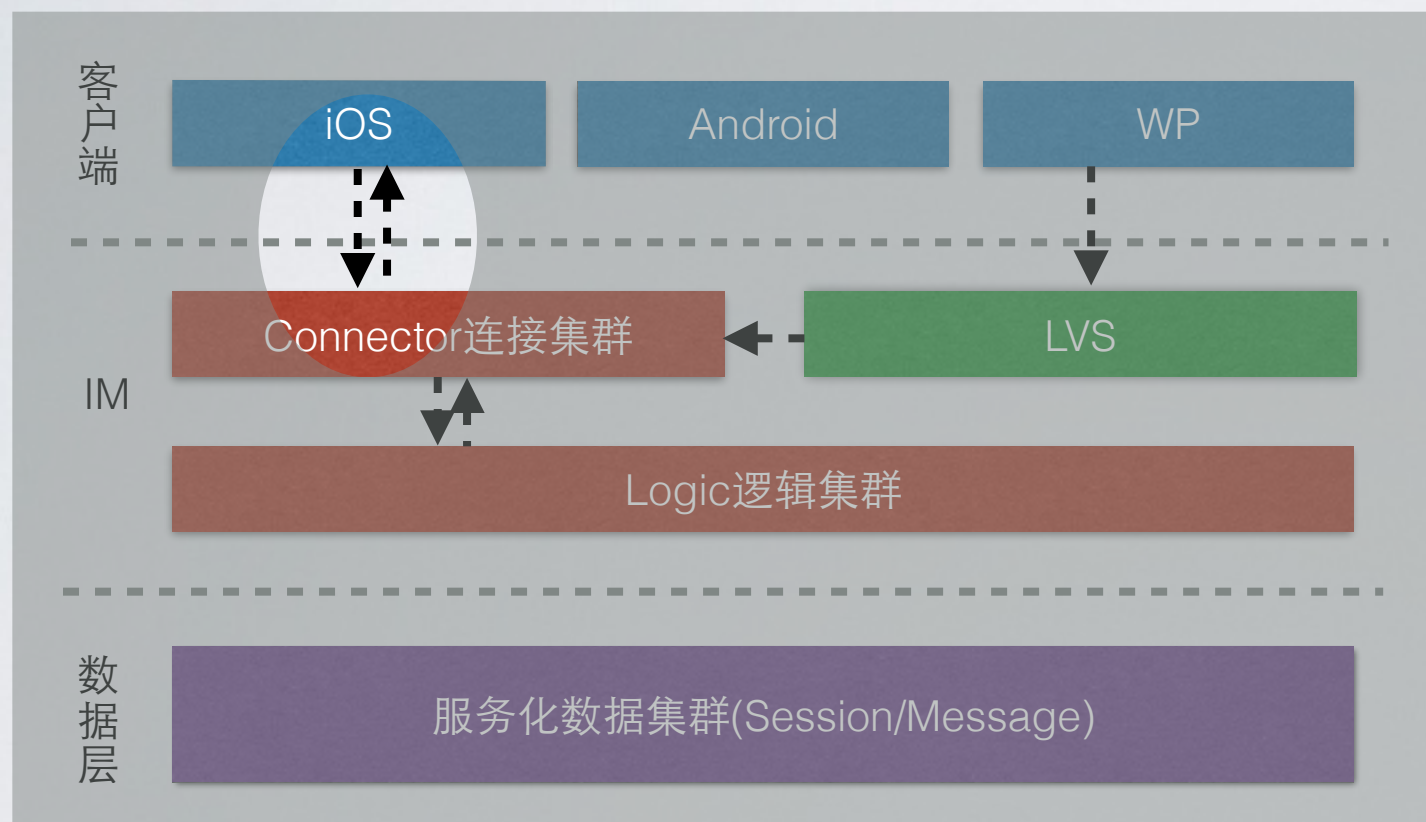


# 逻辑层



1. 用户会话验证
2. 消息存取
3. 异步队列
4. 随时重启

# 通讯协议



“  
安全性要求  
流量要求  
传输要求可靠&高效  
”

# 通讯协议

- 常见协议XMPP/SIP
- 缺点： 1.流量大 2.不可靠 3.交互复杂



# 通信协议设计

目标：

- 高效： 弱网络快速的收发
- 可靠： 不会丢消息
- 易于扩展



# 协议格式

msg:

Flag	Length	Data ...
Flag	Length	Data ...
Flag	Length	Data ...

良好的协议可以：

- 简化系统设计
- 提供可靠个高效的消息传输
- 易于扩展需求

# REDIS协议

## Redis协议

*	number of arguments	CR LF		
\$	bytes of argument 1	CR LF	data	CR LF
\$	bytes of argument 2	CR LF	data	CR LF
\$	bytes of argument 3	CR LF	data	CR LF

SET name latermoon

\* 3 \$ 3 SET \$ 4 name \$ 9 latermoon

下面都用Redis协议来描述逻辑

# READ REDIS COMMAND

```
• 1 func (s *Session) ReadCommand() (cmd *Command, err error) {  
2     // Read ( *<number of arguments> CR LF )  
3     err = s.skipByte('*')  
4     if err != nil { // io.EOF  
5         return  
6     }  
7     // number of arguments  
8     var argCount int  
9     if argCount, err = s.readInt(); err != nil {  
10        return  
11    }  
12    args := make([][]byte, argCount)  
13    for i := 0; i < argCount; i++ {  
14        // Read ( $<number of bytes of argument 1>  
15        err = s.skipByte('$')  
16        if err != nil {  
17            return  
18        }  
19  
20        var argSize int  
21        argSize, err = s.readInt()  
22        if err != nil {  
23            return  
24        }  
25
```

```
26        // Read ( <argument data> CR LF )  
27        args[i] = make([]byte, argSize)  
28        _, err = io.ReadFull(s, args[i])  
29        if err != nil {  
30            return  
31        }  
32  
33        err = s.skipBytes([]byte{CR, LF})  
34        if err != nil {  
35            return  
36        }  
37    }  
38    cmd = NewCommand(args...)  
39    return  
40 }
```

# 基于队列的消息协议

Redis List:

	FIFO
	msg6
	msg5
	msg4
	msg3
	msg2
	msg1

S: msg-send msg1

C: msg-recv 1

S: msg-send msg2

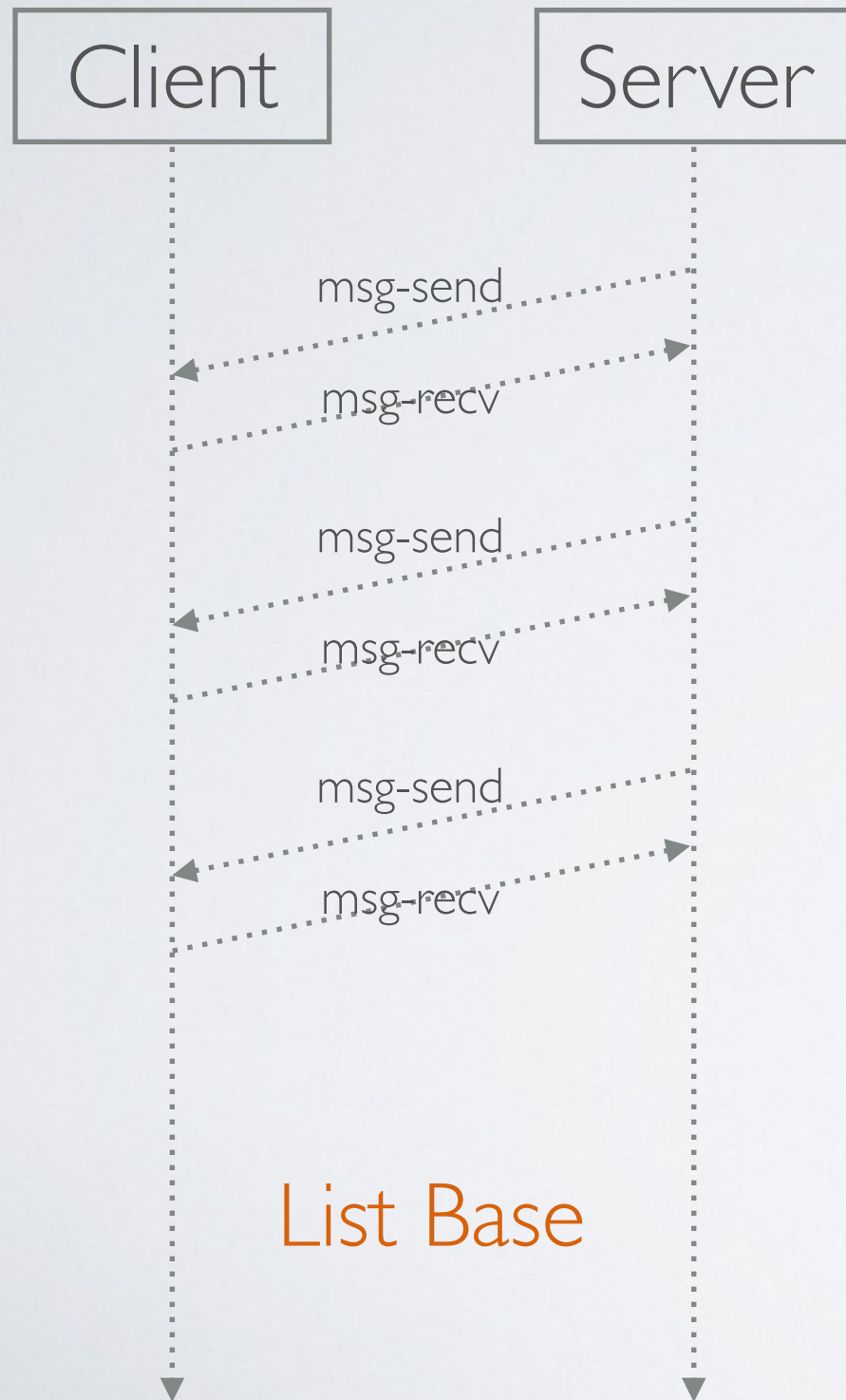
C: msg-recv 2

S: msg-send msg3 msg4 msg 5 msg 6

C: msg-recv 3 4 5 6



# 基于队列的交互



传统的IM协议

前提是基于网线、WIFI，网络延迟极小

移动网络下，交互极其费时，服务器要维护每个状态容易出错

“通讯协议优化，尽量减少一次交互中数据往返的次数。”

# 基于版本号的消息协议

Redis Sorted Set:

Version	Message
106	msg
105	msg
104	msg
103	msg
102	msg
101	msg

S: msg-psh

C: msg-sync

S: msg v 101 msg

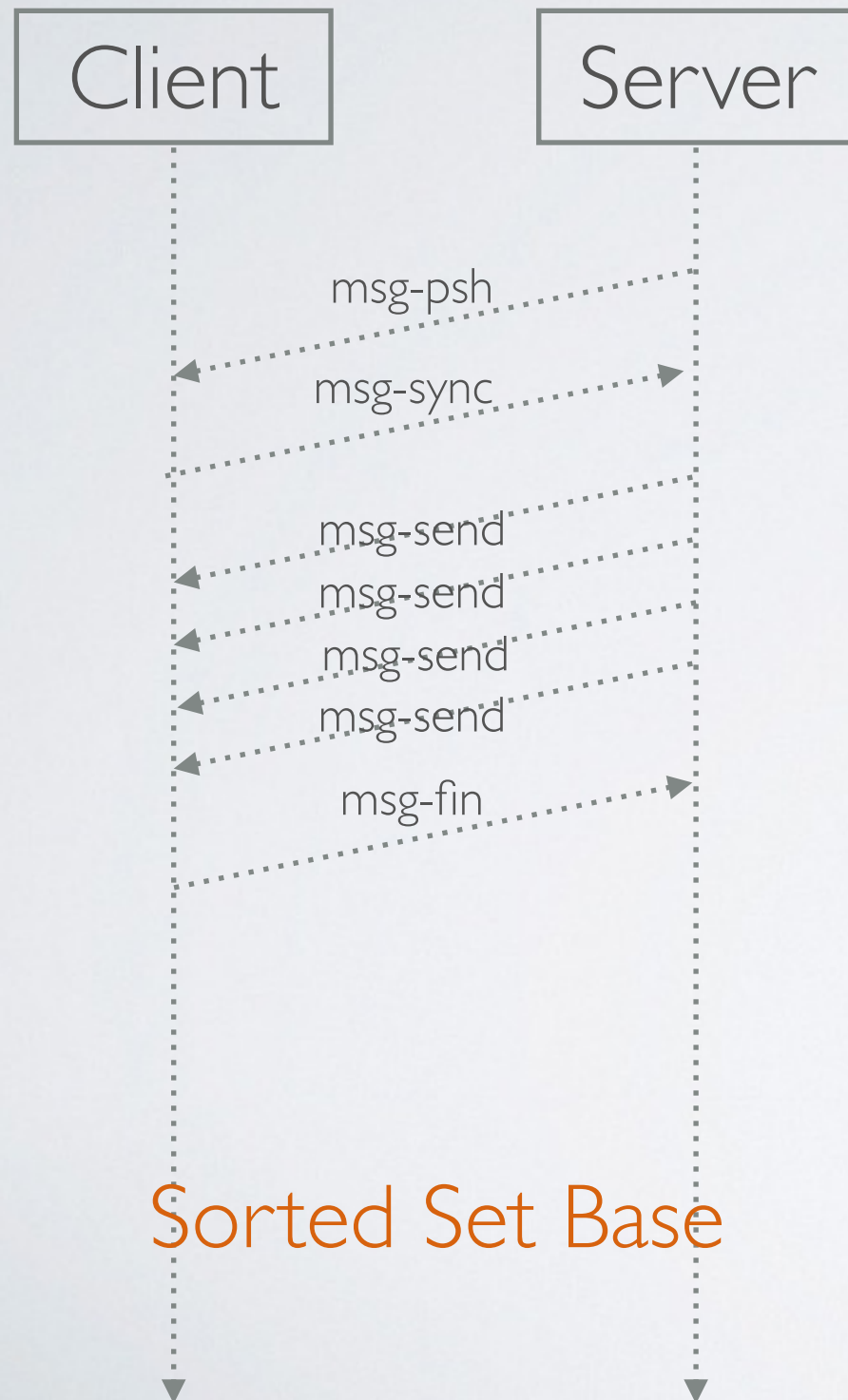
S: msg v 102 msg

S: msg v 103 msg

S: msg v 104 msg

C: msg-fin v 104

# 基于版本号交互



针对弱网络的优化协议


- 消息通过版本号维护顺序
- 新消息到达，Server只负责push通知
- Client收到轻量的msg-psh后发出同步请求
- Server按照版本号连续发送msg
- Client告诉Server收到最后的版本



# 其它问题

- 核心的长连接只用于传输轻量的实时数据
- 图片、语音等都可以开新的TCP或HTTP连接

“一切就绪后，最重要的就是监控。”

	指标名称	当前数值	历史统计曲线图
1	每秒请求量(cmd)		
2	超时量(htime)		
3	队列堆积量(cmdq)		
4	错误量(error)		
--			
5	每秒请求量sum(cmd_sum)		
6	超时量sum(htime_sum)		
7	队列堆积量sum(cmdq_sum)		
8	错误量sum(error_sum)		
--			
9	总用户(tot-ses)		
10	物理连接阶段登陆用户(iop-ses)		
11	逻辑连接阶段待登陆用户(p-ses)		
12	登陆用户(session)		
13	苹果客户端(ios)		
14	英文版苹果客户端(eios)		
15	安卓客户端(android)		
16	微软客户端(wp)		



### < 陌陌后台 IM消息数据统计

通讯实时消息与昨天同时段对比

当前/昨天

上一分钟/昨天

上五分钟/昨天

上十分钟/昨天

上十五分钟/昨天

图片实时消息与昨天同时段对比

当前/昨天

上一分钟/昨天

上五分钟/昨天

上十分钟/昨天

上十五分钟/昨天

监控



“完”

– [latermoon@qq.com](mailto:latermoon@qq.com)