□□□□ Popush □□

清华大学软件学院 司徒静弘

目录

目录

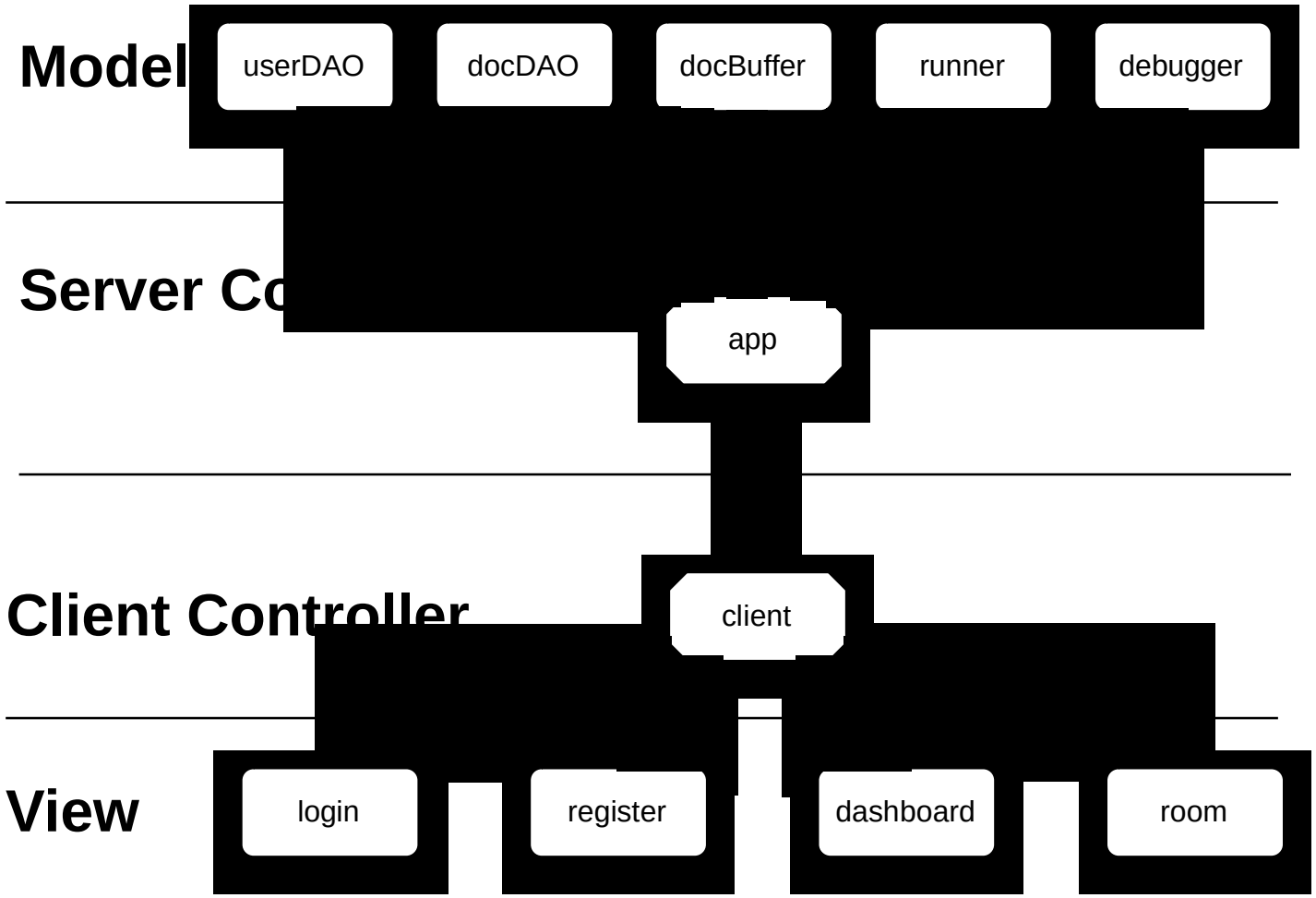
目录

总体设计

MVC 架构

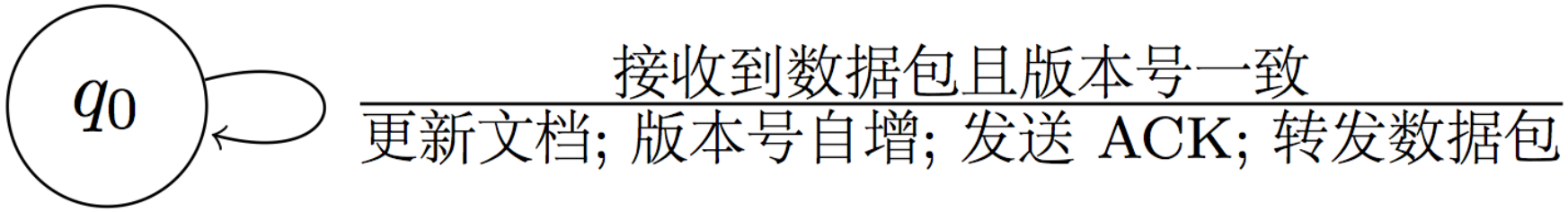
- Model （模型）
  - 程序员编写程序应有的功能（实现算法等等）、数据库专家进行数据管理和数据库设计 (可以实现具体的功能)
- View （视图）
  - 界面设计人员进行图形界面设计
- Controller （控制器）
  - 负责转发请求，对请求进行处理

总体设计


Model
userDAO
docDAO
docBuffer
runner
debugger
Server Co
app
Client Controller
client
View
login
register
dashboard
room

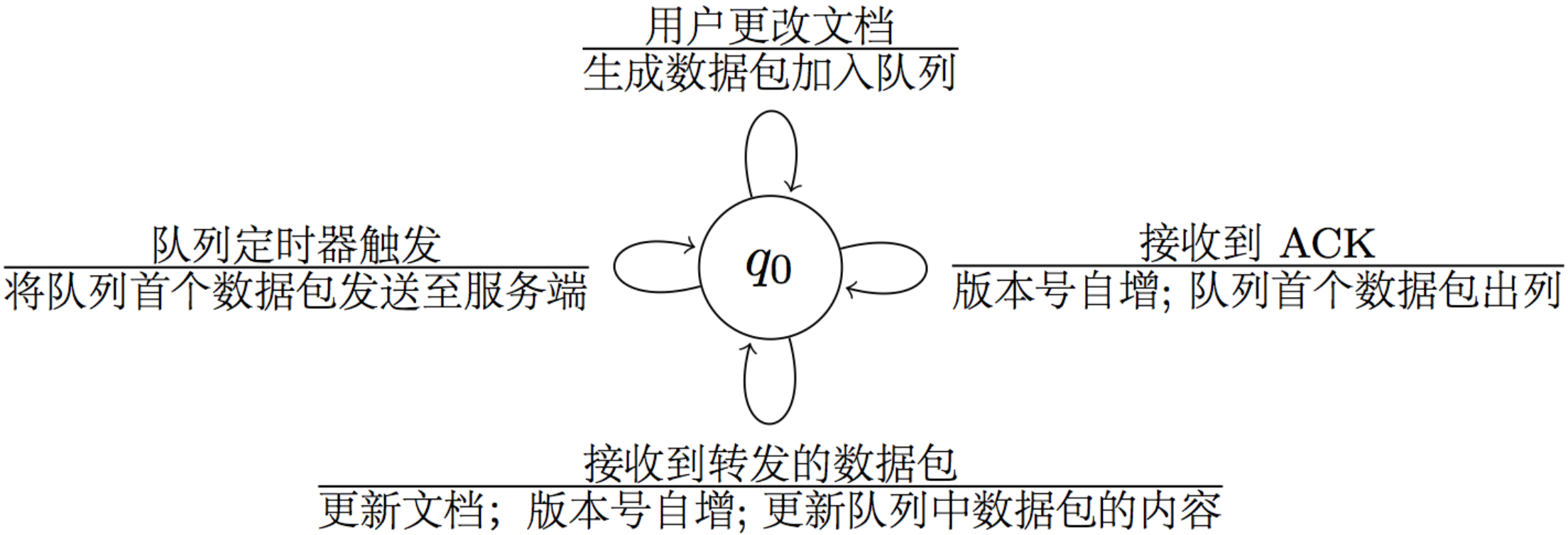
算法设计

后端

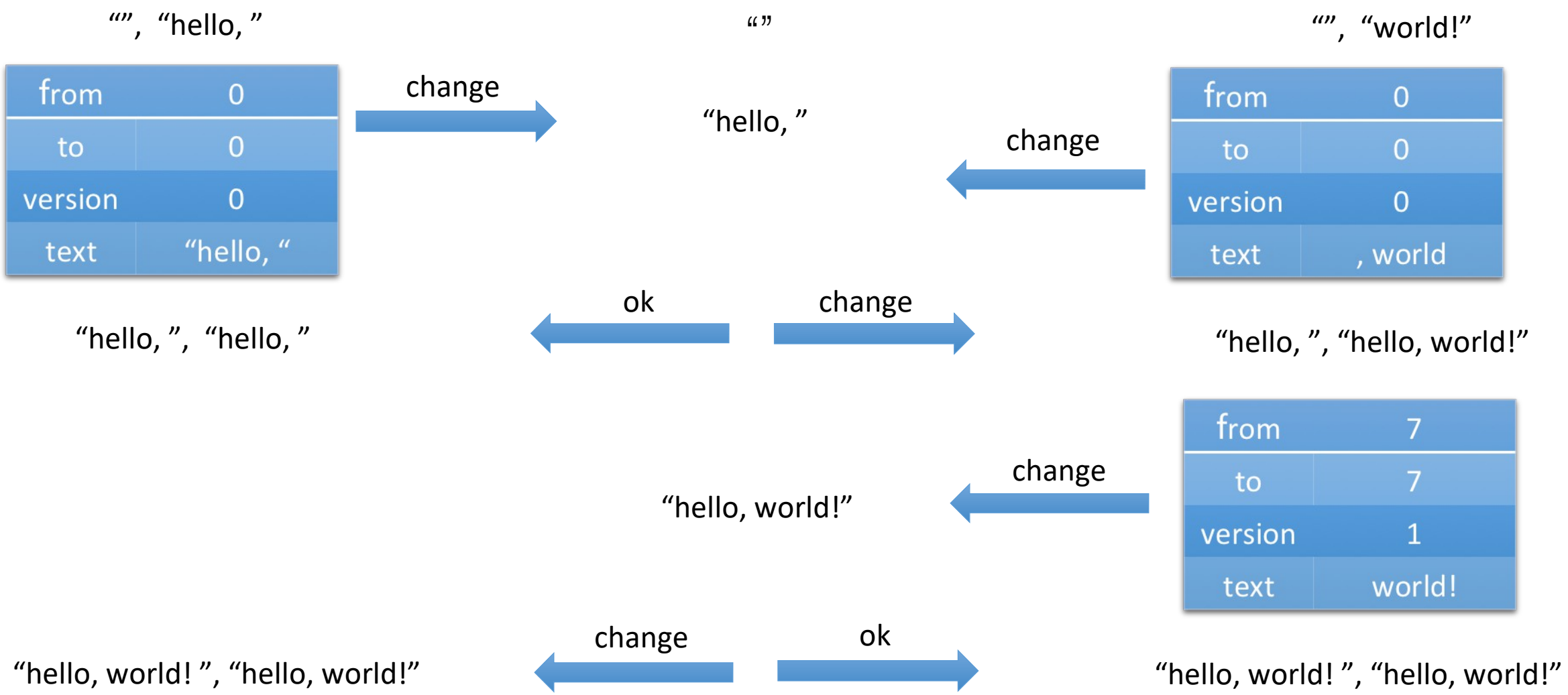


算法设计

前端


用户更改文档
生成数据包加入队列
队列定时器触发
将队列首个数据包发送至服务端
q_0
接收到 ACK
版本号自增；队列首个数据包出列
接收到转发的数据包
更新文档；版本号自增；更新队列中数据包的内容

算法设计


“”, “hello, ”
“”
“”, “world!”
from 0
to 0
version 0
text “hello, “
change
“hello, ”
change
from 0
to 0
version 0
text , world
ok
change
“hello, ”, “hello, ”
“hello, ”, “hello, world!”
from 7
to 7
version 1
text world!
change
“hello, world!”
change
“hello, world! ”, “hello, world!”
change
ok
“hello, world! ”, “hello, world!”

一些技术

Node.js， 事件驱动、非阻塞 I/O， 轻量级、高并发

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
console.log('Server running at http://127.0.0.1:1337/');
```

一些技术

MongoDB，面向文档的 NoSQL 数据库，极其易用

```
db.user.insert({
    name: "akirast",
    password: "123456",
    avatar: "https://popush.com
        /faces/1372495592245.png",
    createTime: new Date().getTime(),
},
```

一些技术

Socket.io ， WebSocket (HTTP/1.1) 的一种实现

```
//client
socket.emit('login', {
    name:$('#login-inputName').val(),
    password:$('#login-inputPassword').val()
});
```

```
// server
socket.on('login', function(data){
    // do something
}
```

一些技术

Nginx，轻量级高性能 HTTP 服务器，静态文件、 WebSocket 转发

```
server {
    listen 80;
    server_name _;

    error_page 403 404 @404;

    location @404 {
        rewrite .* / permanent;
    }

    location / {
        root   /popush/static;
        index  index.html;
    }

    location /socket {
        proxy_pass http://localhost:4444;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Real_IP $remote_addr;
    }
}
```

目录

方式 1

直接改前端， MVC 化

- 封装各种 Controller
- 页面与数据分离，提取 View
- 在通信协议之上构建 Model
- 可自行实现，或使用如 backbone.js 之类的框架
  - 搜索关键字： Javascript MVC Framework

方式 2

根据通信协议重写前端

- 可重新设计前端，包括架构、页面、交互方式等等
- 有风险，大神可一试

Bonus

改进前后端

- 找出后端隐藏很深的一些 BUG （已知）并修复
- 改进算法（包括前后端），以提高响应速度，或使前端在网络极差的环境下也能迅速响应。
- 尝试多进程、分布式
- 提高可靠性（自行想象）

运行与部署

运行与部署是不同的!

- Github 上的 README 是部署方式!
- 如何在本机简单运行（Linux / Mac OS X / Windows）
  - 确保正确安装 Mongodb， Nginx， Node.js
  - 修改 Nginx 配置文件
  - etc……

谢谢大家！

THANKS