

# Network

## Network

- [\[ Doc \]](#) Net ( )
- [\[ Doc \]](#) UDP/Datagram
- [\[ Doc \]](#) HTTP
- [\[ Doc \]](#) DNS ( )
- [\[ Doc \]](#) ZLIB ( )
- [\[ Point \]](#) RPC

## Net

TCP/IP , TCP/IP , 1 , TCP/IP Wireshark . W.Richard Stevens

, TCP `socket.bufferSize`), . IO . , (

, , , , .

, send data1 data2, :

- A. data1, data2 .
- B. data1 , data1 data2 .
- C. data1 data2 , data2 .
- D. data1 data2 .

BCD . , :

- 2. Nagle
- 3. /

1

send , . , .

2

Nagle , Node `socket.setNoDelay()` Nagle , send .  
( ), . , , Nagle .  
, , Nagle , , TCP recv,

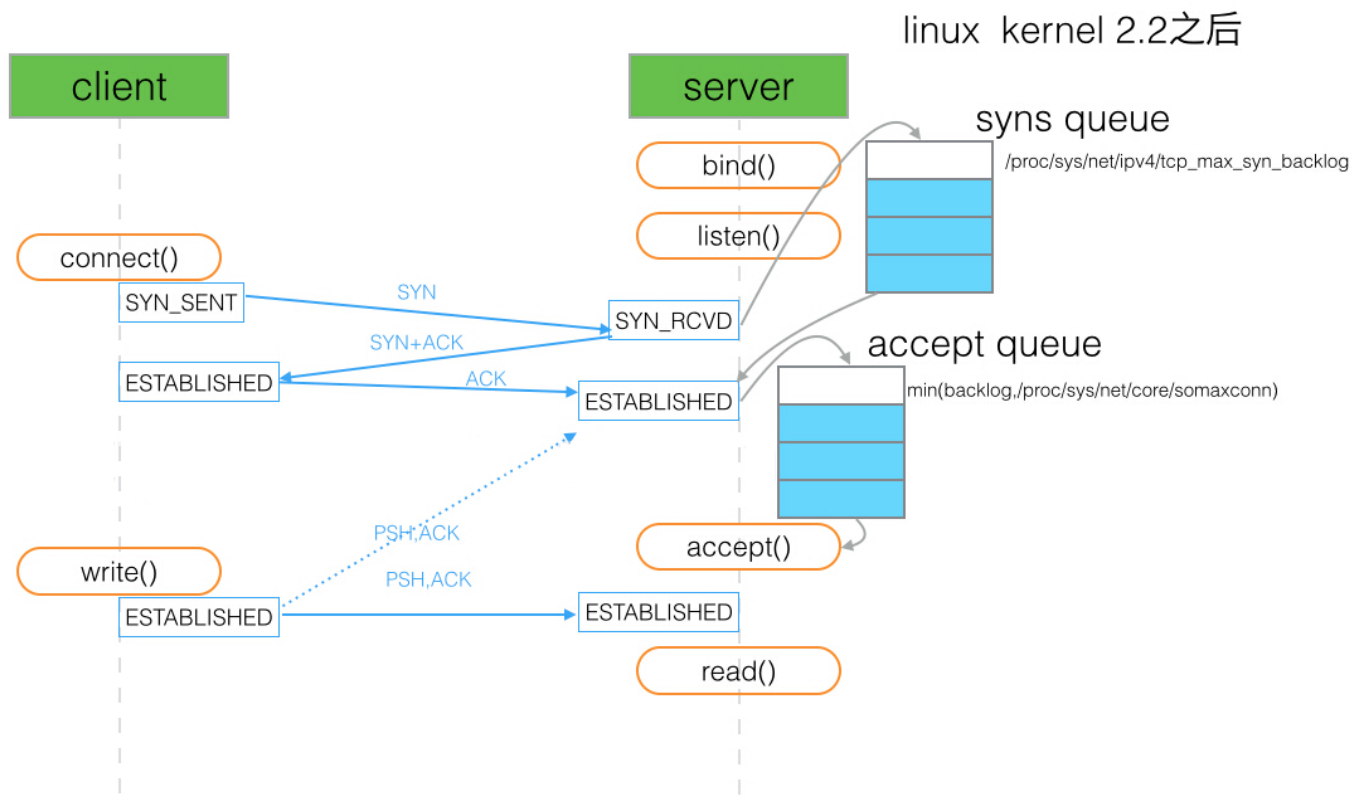
3

/ . , / , .  
(SYN, Synchronize packet), (ACK, Acknowledgement)TCP

## window

TCP Window TCP .  
“ window ?  
, , . TCP window window .

## backlog

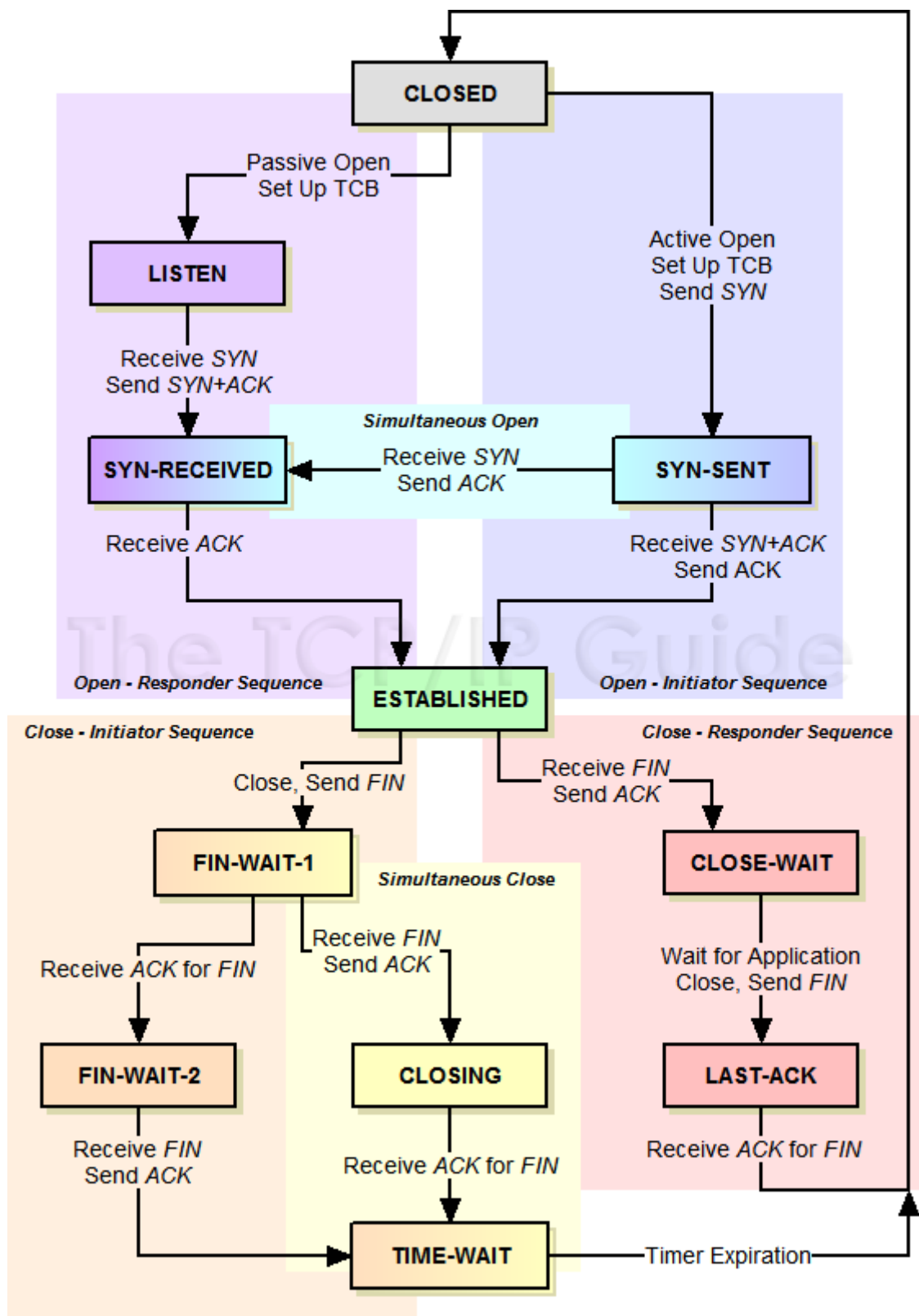


backlog :

“ The behavior of the backlog argument on TCP sockets changed with Linux 2.2. Now it specifies the queue length for completely established sockets waiting to be accepted, instead of the number of incomplete connection requests.

backlog ESTABLISHED accept ( accept queue). backlog , accept qu  
5000 , 13s, nginx , PHP accept backlog DEFALTE 511  
backlog 511.

, backlog somaxconn accept queue tcp\_max\_syn\_backlog SYN queue .



The TCP/IP Guide

state	
CLOSED	,
LISTEN	, SYN
SYN-SENT	SYN,

state	
SYN-RECEIVED	SYN, ACK
ESTABLISHED	SYN-RECEIVED ACK, .
CLOSE-WAIT	(FIN), ACK, , , FIN. LA
LAST-ACK	ACK CLOSE-WAIT FIN, CLOSED
FIN-WAIT-1	FIN, ACK
FIN-WAIT-2	ACK, FIN
CLOSING	FIN, FIN-WAIT-1 ACK, ACK
TIME-WAIT	FIN, FIN ACK, ACK, FIN,

“ TIME\_WAIT TIME\_WAIT ?

TIME\_WAIT ( prevent potential overlap with new connections see (FIN) [TCP Connection Termination](#) for more).

TIME\_WAIT , , keepAlive. keepAlive, TIME\_WAIT CLOSE\_WAIT CLOSE\_WAIT,

## UDP

“ TCP/UDP ? UDP ?

TCP	(Connection oriented)	(1:1)	( )	( SYN )	,				20~60
UDP	(Connection less)	n:m	( )		,				8

UDP socket n m dgram.createSocket(options[, callback]) option reuseAddr SO\_REUSEADDR . SO\_REUSEADDR n m ( ).

TCP		SMTP
		TELNET
		SSH
		HTTP

	FTP	
UDP		DNS
		TFTP
		NTP
		NFS
		RIP
	IP	-
		-

UDP, UDP, / Stats UDP. Node.js UDP

# HTTP

(http servers) HTTP, 2-3 HTTP, RESTful, RESTful URI (Resources), method, status

## method/status

HTTP (method) (status), Node.js, .

```
const http = require('http');

console.log(http.METHODS);
console.log(http.STATUS_CODES);
```

method, method using RESTful Methods for RESTful Services

methods	CRUD		
GET	Read	✓	✓
POST	Create		
PUT	Update/Replace	✓	
PATCH	Update/Modify		
DELETE	Delete	✓	

```
“ GET POST ?
```

, , url . , GET POST ( , ) ,  
, , | . RESTful

```
“ POST PUT ?
```

POST (create) , , POST . PUT Update/Replace, , PUT .

## headers

HTTP headers HTTP . (Request) , .

- [Request fields](#)
- [Response fields](#)

```
“ cookie session ? cookie?
```

, session , cookie . session cookie . cookie ( ). cookie

```
“ ? ?
```

, XMLHttpRequest Fetch HTTP , host (host = hostname : port)  
request). [CORS headers](#) `Access-Control-Allow-` . :

```
location ~* ^/(?:v1|_) {  
    if ($request_method = OPTIONS) { return 200 ''; }  
    header_filter_by_lua '  
        ngx.header["Access-Control-Allow-Origin"] = ngx.var.http_origin; #  
        ngx.header["Access-Control-Allow-Methods"] = "GET, POST, PUT, DELETE, PATCH, OPTIONS";  
        ngx.header["Access-Control-Allow-Credentials"] = "true";  
        ngx.header["Access-Control-Allow-Headers"] = "Content-Type";  
    ';  
    proxy_pass http://localhost:3001;
```

```
}
```

[1] host . host , HTTP HTTPS ,

```
“Script error.” ? ?
```

, (CORS), js Script error. Access-Control-Allow-Origin crossorigin :

```
<script src="http://another-domain.com/app.js" crossorigin="anonymous"></script>
```

JavaScript Script Error.

## Agent

Node.js http.Agent HTTP socket (pooling sockets used in HTTP client requests). H1  
Agent http.globalAgent.

, Node.js 6.8.1 6.10 :

- 1. keepAlive true , socket
- 2. keepAlive true @zcs19871221 socket()

1 2 , request timeout, socket , , , req,

issue Node bagmit, issue repo

## socket hang up

hang up , socket hang up socket . Node.js response , socket "

Node.js :

```
function socketCloseListener() {  
  var socket = this;  
  var req = socket._httpMessage;  
  
  // Pull through final chunk, if anything is buffered.  
  // the ondata function will handle it properly, and this
```

```

// is a no-op if no final chunk remains.
socket.read();

// NOTE: It's important to get parser here, because it could be freed by
// the `socketOnData`.
var parser = socket.parser;
req.emit('close');
if (req.res && req.res.readable) {
  // Socket closed before we emitted 'end' below.
  req.res.emit('aborted');
  var res = req.res;
  res.on('end', function() {
    res.emit('close');
  });
  res.push(null);
} else if (!req.res && !req.socket._hadError) {
  // This socket error fired before we started to
  // receive a response. The error needs to
  // fire on the request.
  req.emit('error', createHangUpError()); // <----- socket hang up
  req.socket._hadError = true;
}

// Too bad. That output wasn't getting written.
// This is pretty terrible that it doesn't raise an error.
// Fixed better in v0.10
if (req.output)
  req.output.length = 0;
if (req.outputEncodings)
  req.outputEncodings.length = 0;

if (parser) {
  parser.finish();
  freeParser(parser, req, socket);
}
}

```

, , F5 . , throw socket hang up.

# DNS

TCP/IP , Alan -> 192.168.0.11 , , , ip (Domain name),

DNS UDP, Node.js :

.lookup(hostname[, options], cb)	/etc/hosts)			
.resolve(hostname[, rrtype], cb)	DNS (rrtype )			

“ DNS .lookup .resolve ?

ip getaddrinfo , , hosts , .lookup hosts , .resolve .lookup getaddrinfo . Node ,

“ hosts ? DNS ?

hosts , IP “ ”, , hosts IP , IP . , IP , , => hosts , => DNS

# ZLIB

, , , . zlib Gzip/Gunzip, Deflate/Inflate Deflat

TODO

# RPC

RPC (Remote Procedure Call Protocol) TCP/IP Node.js http . , (

RPC :

- Thrift
- HTTP
- MQ

# Thrift

“Thrift” RPC Facebook “C#C++POSIX” Cappuccino DelphiErlangGoHaskellJava  
Node.jsOCamlPerlPHPPythonRubySmalltalk Facebook  
2007 4 Facebook Apache

# HTTP

“ gRPC is an open source remote procedure call (RPC) system initially developed at Google. It uses HTTP/2 for transport, Protocol Buffers as the interface description language, and provides features such as authentication, bidirectional streaming and flow control, blocking or nonblocking bindings, and cancellation and timeouts. It generates cross-platform client and server bindings for many languages.

## MQ

(Message Queue)    RPC    (RPC over mq)    ,    /    /    .

TODO