



Atividade 4: Backlog e Processos

Zumbis

Alunos: Elias Santos Martins e Gabriel Sanders

Pereira Sobral

RA: 247057 e 247118

Instituto de Computação

Universidade Estadual de Campinas

Campinas, 19 de Agosto de 2025.

Sumário

1	Dinâmica das filas de conexão TCP no Kernel Linux	2
1.1	Fila de conexões incompletas (SYN queue) e fila de conexões completas (accept queue)	2
1.2	Parâmetro backlog da função listen()	2
1.3	Parâmetro /proc/sys/net/ipv4/tcp_max_syn_backlog	2
2	Servidor TCP concorrente com fork()	3
2.1	Backlog como argumento de linha de comando	3
2.2	Configuração de tempo de espera (sleep)	3
2.3	Execução sem tratador de sinais para SIGCHLD	3
2.4	Implementação do tratador de sinais para SIGCHLD	3
2.5	Explicação sobre processos zumbis	5
3	Automatização de testes de carga com script	5
3.1	Execução com diferentes valores de backlog	5
3.2	Tabela de resultados	6
3.3	Comparação com /proc/sys/net/ipv4/tcp_max_syn_backlog	6
4	Análise de pacotes com sniffer	7
4.1	Flags TCP observadas em clientes sem conexão	7

1 Dinâmica das filas de conexão TCP no Kernel Linux

1.1 Fila de conexões incompletas (SYN queue) e fila de conexões completas (accept queue)

No Kernel Linux, temos duas filas de conexões:

- **Fila de conexões incompletas:** fila de conexões que foram solicitadas por clientes, mas cujo 3 way handshake ainda não está completo. Uma conexão é colocada nessa fila ao se receber o primeiro sinal SYN de um cliente
- **Fila de conexões completas:** fila de conexões cujo 3 way handshake foi completo. Uma conexão é movida da fila de conexões incompletas para esta fila ao se receber o primeiro ACK do cliente, que finaliza o 3 way handshake.

1.2 Parâmetro backlog da função listen()

O parâmetro backlog determina o tamanho máximo das filas de conexão, de modo que, quando a soma das duas filas ultrapassar o backlog, o servidor simplesmente ignora o SYN.

1.3 Parâmetro `/proc/sys/net/ipv4/tcp_max_syn_backlog`

A partir do Linux versão 2.2, o parâmetro *backlog* da função *listen* passou a definir somente o tamanho máximo da fila de conexões completas (<https://linux.die.net/man/2/listen>). O parâmetro `/proc/sys/net/ipv4/tcp_max_syn_backlog`

pertence ao kernel e define o tamanho máximo da fila de conexões incompletas.

2 Servidor TCP concorrente com `fork()`

2.1 Backlog como argumento de linha de comando

Implementamos o backlog como o primeiro parâmetro da execução do servidor: .

```
servidor <backlog>
```

2.2 Configuração de tempo de espera (`sleep`)

Implementamos o backlog como o segundo parâmetro da execução do servidor: .

```
servidor <backlog> <tempo_sleep>
```

2.3 Execução sem tratador de sinais para SIGCHLD

Executamos o servidor e vários clientes, sem o tratamento de sinal. Podemos ver na figura 1 os clientes zumbis

2.4 Implementação do tratador de sinais para SIGCHLD

Implementamos o tratador de sinais para SIGCHLD no servidor, e fizemos o mesmo procedimento de executar vários clientes em sequência. Podemos ver agora na listagem do *ps* que não há processos zumbis.

```
saanders@saanders-Inspiron-7591-2n1: ~/Desktop/MC833/Lab02
saanders@saanders-Inspiron-7591-2n1:~/Desktop/MC833/Lab02$ ps aux | grep servidor
saanders 26479 0.0 0.0 2780 1536 pts/1 S+ 20:27 0:00 ./servidor 7777 7 7
saanders 26597 0.0 0.0 2780 896 pts/1 S+ 20:28 0:00 ./servidor 7777 7 7
saanders 26601 0.0 0.0 2780 896 pts/1 S+ 20:28 0:00 ./servidor 7777 7 7
saanders 26617 0.0 0.0 2780 896 pts/1 S+ 20:28 0:00 ./servidor 7777 7 7
saanders 26638 0.0 0.0 9220 2560 pts/2 S+ 20:28 0:00 grep --color=auto servidor
saanders@saanders-Inspiron-7591-2n1:~/Desktop/MC833/Lab02$ ps aux | grep servidor
saanders 26479 0.0 0.0 2780 1536 pts/1 S+ 20:27 0:00 ./servidor 7777 7 7
saanders 26597 0.0 0.0 0 0 pts/1 Z+ 20:28 0:00 [servidor] <defunct>
saanders 26601 0.0 0.0 0 0 pts/1 Z+ 20:28 0:00 [servidor] <defunct>
saanders 26617 0.0 0.0 0 0 pts/1 Z+ 20:28 0:00 [servidor] <defunct>
saanders 26685 0.0 0.0 9220 2560 pts/2 S+ 20:28 0:00 grep --color=auto servidor
saanders@saanders-Inspiron-7591-2n1:~/Desktop/MC833/Lab02$

HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 91
Connection: close

<html><head><title>MC833</title></head><body><h1>MC833 TCP Concorrente </h1></body>
</html>
saanders@saanders-Inspiron-7591-2n1:~/Desktop/MC833/Lab02$ ./cliente 127.0.0.1 7777
7
local IP/port: 127.0.0.1:53574
remote IP/port: 127.0.0.1:7777
HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 91
Connection: close

<html><head><title>MC833</title></head><body><h1>MC833 TCP Concorrente </h1></body>
</html>
saanders@saanders-Inspiron-7591-2n1:~/Desktop/MC833/Lab02$
```

Figura 1: Processos zumbis gerados por clientes encerrados

```
saanders@saanders-Inspiron-7591-2n1: ~/Desktop/MC833/Lab02
saanders@saanders-Inspiron-7591-2n1:~/Desktop/MC833/Lab02$ ps aux | grep servidor
saanders 27287 0.0 0.0 2780 1536 pts/1 S+ 20:50 0:00 ./servidor 7777 7 7
saanders 27515 0.0 0.0 9220 2560 pts/2 S+ 20:50 0:00 grep --color=auto servidor
saanders@saanders-Inspiron-7591-2n1:~/Desktop/MC833/Lab02$

HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 91
Connection: close

<html><head><title>MC833</title></head><body><h1>MC833 TCP Concorrente </h1></body>
</html>
saanders@saanders-Inspiron-7591-2n1:~/Desktop/MC833/Lab02$ ./cliente 127.0.0.1 7777
7
local IP/port: 127.0.0.1:49116
remote IP/port: 127.0.0.1:7777
HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 91
Connection: close

<html><head><title>MC833</title></head><body><h1>MC833 TCP Concorrente </h1></body>
</html>
saanders@saanders-Inspiron-7591-2n1:~/Desktop/MC833/Lab02$
```

Figura 2: Ausência de processos zumbis após tratamento do SIGCHLD pelo servidor

2.5 Explicação sobre processos zumbis

Um processo zumbi é um processo que já foi encerrado, mas cujo pai ainda não recebeu o sinal de retorno. Portanto, é mantido uma entrada na tabela de processos para esse processo, que fica marcado com o estado Z (zumbi). O processo só será removido dessa tabela quando o processo pai correspondente chamar o *wait()*. Se o pai não está configurado para receber a notificação de encerramento do processo filho, o processo filho permanecerá na tabela, no estado de zumbi.

3 Automatização de testes de carga com script

3.1 Execução com diferentes valores de backlog

Para lancar vários clientes simultâneos em diferentes valores de backlog, implementamos um script bash, que executa um servidor para cada valor de backlog indo de 0 até 10. Para cada servidor, rodamos o comando `seq 10 | xargs -P10 -I ./cliente 127.0.0.1 7777` para executar 10 clientes simultaneamente. Colocamos um sleep de 5 segundos no cliente antes de ele fechar a conexão, para ser possível capturar a conexão no estado de ESTABLISHED.

3.2 Tabela de resultados

Tabela 1: Resultados da variação do backlog no servidor TCP

Backlog	Conexões imediatas	Conexões rejeitadas
0	10	0
1	3	0
2	4	0
3	5	0
4	6	0
5	7	0
6	8	0
7	9	0
8	10	0
9	10	0
10	10	0

Podemos ver que o valor de conexões imediatas não bate exatamente com o valor do backlog, mas é proporcional (sempre 2 a mais).

3.3 Comparação com `/proc/sys/net/ipv4/tcp_max_syn_backlog`

O parâmetro `backlog` cuida do tamanho da fila de conexões completas, enquanto o `tcp_max_syn_backlog` da fila de conexões incompletas. No nosso experimento alteramos apenas o tamanho da fila de conexões completas, e a fila de conexões incompletas ficou com seu tamanho padrão

```

00:00:00.000233 IP 127.0.0.1.50410 > 127.0.0.1.7777: Flags [S], seq 1601761500, win 65495, options [mss 65495,sackOK,TS val 1222408445 ecr 0,nop,wscale 7], length 0
00:00:00.000219 IP 127.0.0.1.50418 > 127.0.0.1.7777: Flags [S], seq 379807735, win 65495, options [mss 65495,sackOK,TS val 1222408445 ecr 0,nop,wscale 7], length 0
00:00:00.000205 IP 127.0.0.1.50428 > 127.0.0.1.7777: Flags [S], seq 4067752364, win 65495, options [mss 65495,sackOK,TS val 1222408445 ecr 0,nop,wscale 7], length 0
00:00:00.000369 IP 127.0.0.1.50444 > 127.0.0.1.7777: Flags [S], seq 3465434996, win 65495, options [mss 65495,sackOK,TS val 1222408446 ecr 0,nop,wscale 7], length 0
00:00:01.043778 IP 127.0.0.1.50392 > 127.0.0.1.7777: Flags [S], seq 1642466276, win 65495, options [mss 65495,sackOK,TS val 1222409490 ecr 0,nop,wscale 7], length 0
00:00:00.000056 IP 127.0.0.1.50428 > 127.0.0.1.7777: Flags [S], seq 4067752364, win 65495, options [mss 65495,sackOK,TS val 1222409490 ecr 0,nop,wscale 7], length 0
00:00:00.000001 IP 127.0.0.1.50444 > 127.0.0.1.7777: Flags [S], seq 3465434996, win 65495, options [mss 65495,sackOK,TS val 1222409490 ecr 0,nop,wscale 7], length 0
00:00:00.000002 IP 127.0.0.1.50410 > 127.0.0.1.7777: Flags [S], seq 1601761500, win 65495, options [mss 65495,sackOK,TS val 1222409490 ecr 0,nop,wscale 7], length 0
00:00:00.000001 IP 127.0.0.1.50396 > 127.0.0.1.7777: Flags [S], seq 82711555, win 65495, options [mss 65495,sackOK,TS val 1222409490 ecr 0,nop,wscale 7], length 0
00:00:00.000001 IP 127.0.0.1.50418 > 127.0.0.1.7777: Flags [S], seq 379807735, win 65495, options [mss 65495,sackOK,TS val 1222409490 ecr 0,nop,wscale 7], length 0
00:00:00.000003 IP 127.0.0.1.50404 > 127.0.0.1.7777: Flags [S], seq 1386369183, win 65495, options [mss 65495,sackOK,TS val 1222409490 ecr 0,nop,wscale 7], length 0
00:00:01.023922 IP 127.0.0.1.50404 > 127.0.0.1.7777: Flags [S], seq 1386369183, win 65495, options [mss 65495,sackOK,TS val 1222410514 ecr 0,nop,wscale 7], length 0
00:00:00.000001 IP 127.0.0.1.50396 > 127.0.0.1.7777: Flags [S], seq 82711555, win 65495, options [mss 65495,sackOK,TS val 1222410514 ecr 0,nop,wscale 7], length 0
00:00:00.000012 IP 127.0.0.1.50410 > 127.0.0.1.7777: Flags [S], seq 1601761500, win 65495, options [mss 65495,sackOK,TS val 1222410514 ecr 0,nop,wscale 7], length 0
00:00:00.000000 IP 127.0.0.1.50444 > 127.0.0.1.7777: Flags [S], seq 3465434996, win 65495, options [mss 65495,sackOK,TS val 1222410514 ecr 0,nop,wscale 7], length 0
00:00:00.000000 IP 127.0.0.1.50418 > 127.0.0.1.7777: Flags [S], seq 379807735, win 65495, options [mss 65495,sackOK,TS val 1222410514 ecr 0,nop,wscale 7], length 0
00:00:00.000001 IP 127.0.0.1.50428 > 127.0.0.1.7777: Flags [S], seq 4067752364, win 65495, options [mss 65495,sackOK,TS val 1222410514 ecr 0,nop,wscale 7], length 0
00:00:00.000995 IP 127.0.0.1.50392 > 127.0.0.1.7777: Flags [S], seq 1642466276, win 65495, options [mss 65495,sackOK,TS val 1222410515 ecr 0,nop,wscale 7], length 0
00:00:00.929239 IP 127.0.0.1.7777 > 127.0.0.1.50372: Flags [R.], seq 176, ack 4098, win 797, options [nop,nop,TS val 1222411444 ecr 1222408444], length 0
00:00:00.000044 IP 127.0.0.1.7777 > 127.0.0.1.50380: Flags [P.], seq 1:176, ack 4098, win 797, options [nop,nop,TS val 1222411444 ecr 1222408444], length 175
00:00:00.000016 IP 127.0.0.1.50380 > 127.0.0.1.7777: Flags [.], ack 176, win 511, options [nop,nop,TS val 1222411444 ecr 1222411444], length 0
00:00:00.000322 IP 127.0.0.1.7777 > 127.0.0.1.50384: Flags [P.], seq 1:176, ack 4098, win 797, options [nop,nop,TS val 1222411444 ecr 1222408444], length 175
00:00:00.000008 IP 127.0.0.1.50384 > 127.0.0.1.7777: Flags [.], ack 176, win 511, options [nop,nop,TS val 1222411444 ecr 1222411444], length 0
00:00:00.093388 IP 127.0.0.1.50444 > 127.0.0.1.7777: Flags [S], seq 3465434996, win 65495, options [mss 65495,sackOK,TS val 1222411538 ecr 0,nop,wscale 7], length 0
00:00:00.000001 IP 127.0.0.1.50392 > 127.0.0.1.7777: Flags [S], seq 1642466276, win 65495, options [mss 65495,sackOK,TS val 1222411538 ecr 0,nop,wscale 7], length 0
00:00:00.000009 IP 127.0.0.1.50418 > 127.0.0.1.7777: Flags [S], seq 379807735, win 65495, options [mss 65495,sackOK,TS val 1222411538 ecr 0,nop,wscale 7], length 0
00:00:00.000010 IP 127.0.0.1.50428 > 127.0.0.1.7777: Flags [S], seq 4067752364, win 65495, options [mss 65495,sackOK,TS val 1222411538 ecr 0,nop,wscale 7], length 0
00:00:00.000001 IP 127.0.0.1.7777 > 127.0.0.1.50444: Flags [S.], seq 1108786684, ack 3465434997, win 65483, options [mss 65495,sackOK,TS val 1222411538 ecr 1222411538], length 0
00:00:00.000000 IP 127.0.0.1.50404 > 127.0.0.1.7777: Flags [S], seq 1386369183, win 65495, options [mss 65495,sackOK,TS val 1222411538 ecr 0,nop,wscale 7], length 0
00:00:00.000000 IP 127.0.0.1.7777 > 127.0.0.1.50392: Flags [S.], seq 1054895224, ack 1642466277, win 65483, options [mss 65495,sackOK,TS val 1222411538 ecr 1222411538], length 0
00:00:00.000005 IP 127.0.0.1.7777 > 127.0.0.1.50418: Flags [S.], seq 3074610841, ack 379807736, win 65483, options [mss 65495,sackOK,TS val 1222411538 ecr 1222411538], length 0
00:00:00.000002 IP 127.0.0.1.50410 > 127.0.0.1.7777: Flags [S], seq 1601761500, win 65495, options [mss 65495,sackOK,TS val 1222411538 ecr 0,nop,wscale 7], length 0
00:00:00.000003 IP 127.0.0.1.50396 > 127.0.0.1.7777: Flags [S], seq 82711555, win 65495, options [mss 65495,sackOK,TS val 1222411538 ecr 0,nop,wscale 7], length 0
00:00:00.000003 IP 127.0.0.1.7777 > 127.0.0.1.50428: Flags [S.], seq 1342274880, ack 4067752365, win 65483, options [mss 65495,sackOK,TS val 1222411520 ecr 1222411538], length 0
00:00:00.000002 IP 127.0.0.1.50444 > 127.0.0.1.7777: Flags [.], ack 1, win 512, options [nop,nop,TS val 1222411538 ecr 1222411538], length 0
00:00:00.000000 IP 127.0.0.1.50392 > 127.0.0.1.7777: Flags [.], ack 1, win 512, options [nop,nop,TS val 1222411538 ecr 1222411538], length 0
00:00:00.000000 IP 127.0.0.1.7777 > 127.0.0.1.50404: Flags [S.], seq 1834338792, ack 1386369184, win 65483, options [mss 65495,sackOK,TS val 1222411520 ecr 1222411538], length 0

```

Figura 3: Sniffer durante os testes: *sudo tcpdump -i lo -n -ttt 'tcp and port 7777'*

4 Análise de pacotes com sniffer

4.1 Flags TCP observadas em clientes sem conexão

Quando um cliente não consegue se conectar, então aparece a flag [R.] e [S.], a conexão foi recusada e que uma retransmissão do SYNC está sendo feita, como demonstrado na figura 3. Quando a fila está cheia, aparecem apenas SYNC retransmitidos pois o cliente continua tentando estabelecer uma conexão, e faz isso até que o servidor libere espaço em sua fila.