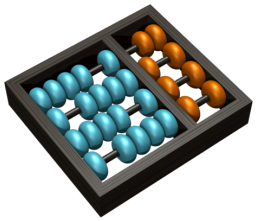


Relatório - Backlog e Processos Zumbis

Alunos: Rian Radeck e Igor Brito

RAs: 187793 e 171929

Instituto de Computação

Universidade Estadual de Campinas

Campinas, 31 de Outubro de 2023.

Sumário

1 Backlog e filas TCP

Segundo o manuel do linux, o argumento `backlog` é utilizado na função `listen` para definir o tamanho máximo da fila de conexões pendentes para o socket que está em modo passivo. Ela é instanciada da seguinte maneira: `int listen(int sockfd, int backlog);`

No protocolo TCP um socket divide o tamanho do seu backlog em duas filas, sendo uma de conexões aceitas (ESTABLISHED state) e outra de conexões incompletas (SYN_RCVD state). Quando uma determinada requisição é processada pelo nosso socket ela é jogada na fila de conexões incompletas e passa para a fila de conexões incompletas a partir do momento que o 3WHS é completado, veja uma simplificação no esquema abaixo:

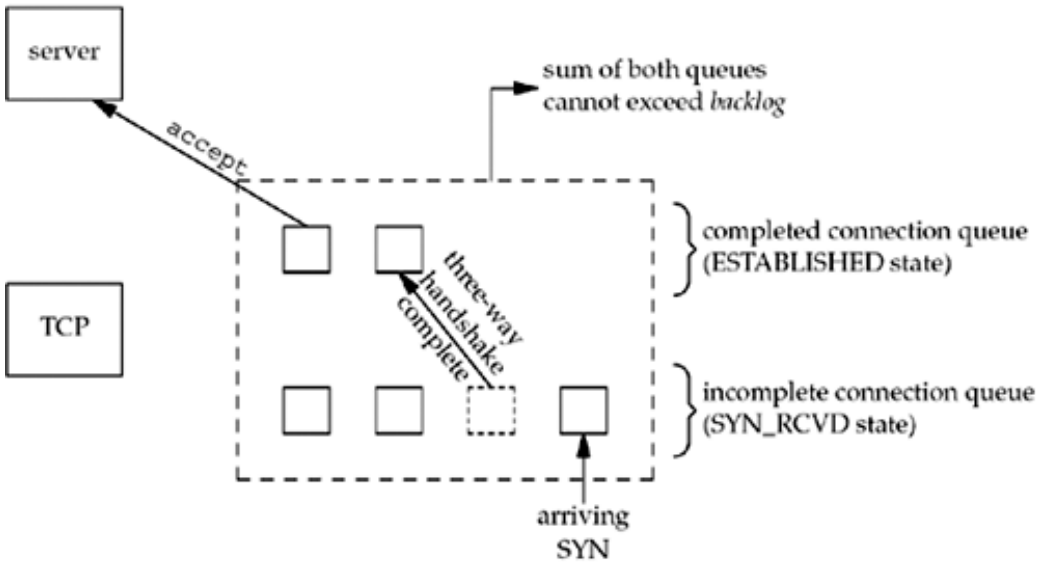
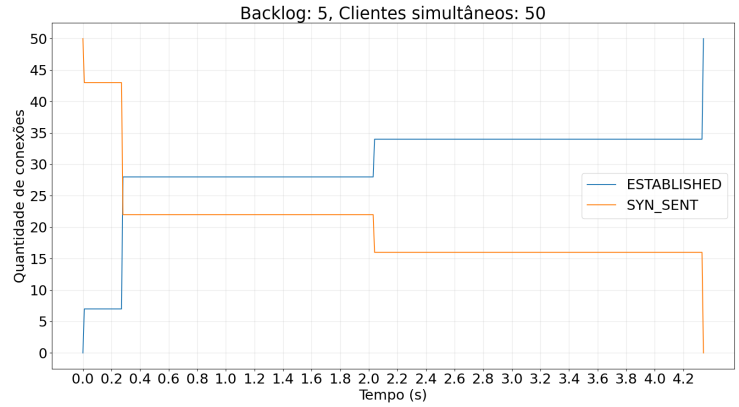
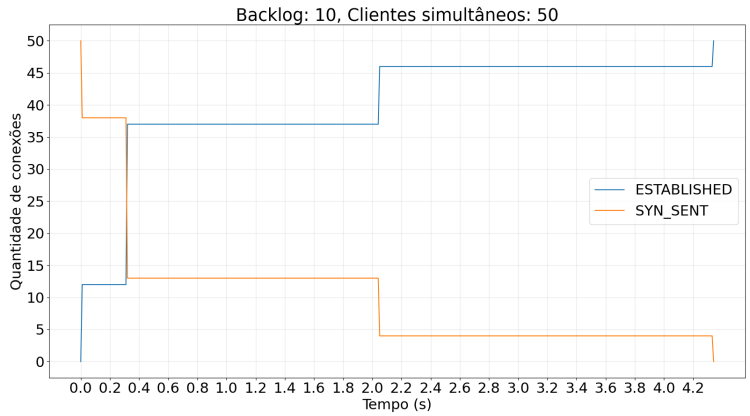


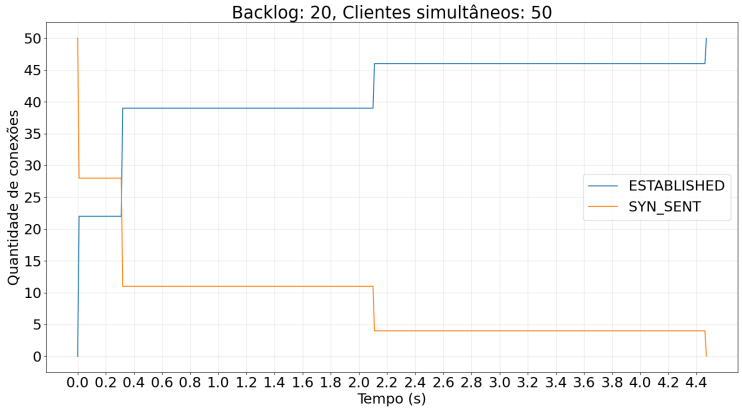
Figura 1: Filas do protocolo TCP para um socket passivo.

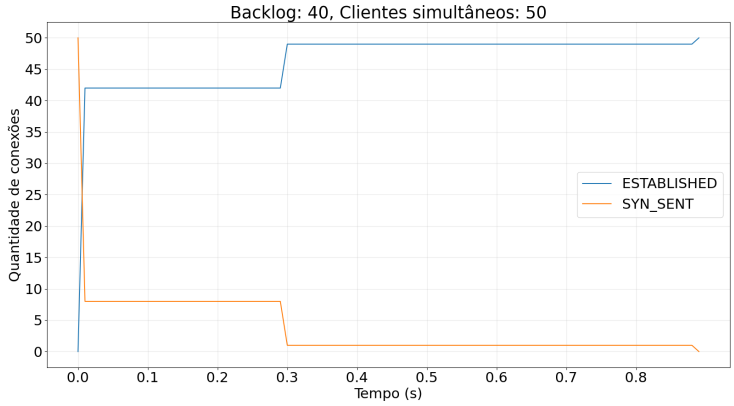
2 Conexões simultâneas

Utilizando o comando `# netstat -taulpn | grep 1234` nós conseguimos observar todo o tráfico de pacotes do nosso servidor (que está rodando na porta 1234. Fizemos um script que inicia vários clientes simultaneamente, monitora a saída desse comando e plota a quantidade de pacotes `ESTABLISHED` e `SYN_SENT` dos clientes que vamos spawnar. O código pode ser encontrado no `script.py`. O último detalhe que é preciso ser notado é que o tempo entre cada `accept` no servidor é de 1s, ou seja, foi colocado um `sleep(1)` no início do loop de aceitação de conexões. Vamos observar alguns resultados que encontramos (tempo de conexão de cada cliente 15s).









O que conseguimos concluir é que na prática o tamanho do backlog afeta principalmente a quantidade de conexões iniciais que serão estabelecidas pelo nosso servidor (mesmo tendo 1s entre os accepts?), mas é válido notar que no manual da chamada `listen` existem muitos *may*'s, ou seja muito é decidido pela máquina em que o código é rodado.

3 Processos zumbis

Processos zumbis são processos que já encerraram sua execução, mas ainda possuem entrada na tabela de processos, pois estão aguardando o

processo pai ler seu `exit status`. Estava de fato ocorrendo a aparição de processos zumbis.

Podemos liberar os recursos do processo filho utilizando um waitpid. Essa função espera por mudanças de estado nos processos filhos do processo que a invoca. Essas mudanças podem ser a finalização do processo ou a sua interrupção por um sinal por exemplo.

Para consertar esse problema iremos instalar um signal handler que tratará os sinais `SIGCHLD` que são emitidos pelos processos filhos para o processo pai.

```
1  void sig_chld_handler(int signo)
2  {
3    pid_t pid;
4    int  stat;
5           //Itera  sobre os  processos  filhos  que   j   finalizaram sem bloquear
6    //  atravs   da flag  WNOHANG
7    while((pid = waitpid(-1, &stat, WNOHANG)) > 0)
8         printf("Child %d terminated\n", pid);
9    return;
10 }
11  ...
12 int  main()
13 {
14  ...
15 //A seguir dentro da main antes de aceitar   conexes   dos  clientes ,
16 //iremos instalar  o  signal  handler:
17 signal(SIGCHLD, sig_chld_handler);
```

4 Performance

Analisamos a quantidade de conexões bem-sucedidas num intervalo de 10 segundos através do script `perf.py`. Invocamos em paralelo 200 instâncias de clientes e segue os resultados variando o backlog no intervalo $[0, 15]$.

backlog	ESTAB	$\text{SYN}_S ENT$
0	62	138
1	51	148
2	73	127
3	78	122
4	66	134
5	51	148
6	48	151
7	74	125
8	79	120
9	61	138
10	43	156
11	86	113
12	23	176
13	63	136
14	39	160
15	47	154

Como podemos observar os resultados não seguem um padrão muito claro e as variações parecem estar dentro de uma margem de erro. Através de uma extensa pesquisa descobrimos que geralmente o valor de backlog é descartado pela maioria dos sistemas operacionais modernos. De acordo

com o manual:

> 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. The maximum length of the queue for incomplete sockets can be set using /proc/sys/net/ipv4/tc_max_syn_backlog. When syncookies are enabled there is no logical maximum length and this setting is ignored. See tcp(7) for more information.

Portanto concluímos que não importa o valor de backlog escolhido, a performance será parecida. Podemos escolher até mesmo 0.