

Part 3, Monitoring your network packets and tuning the network

Article Number: 217 | Rating: Unrated | Last Updated: Mon, Jun 4, 2018 12:20 PM

Optimizing AIX 7 network performance

Part 3, Monitoring your network packets and tuning the network

Martin Brown and Ken Milberg

Published on January 11, 2011

[Facebook](#)[Twitter](#)[Linked In](#)[Google+](#)[E-mail this page](#)



1

Content series:

This content is part of the series: Optimizing AIX 7 network performance

While running commands (such as netstat) can provide useful information, sometimes you need to drill down more to the packet level. This is where tracing tools come in handy, such as iptrace, ipreport, and tcpdump. You'll also learn how you can tune a network using tools such as no. The no command is similar to vmo and ioo, but no is the network flavor. This article focuses on tcp workload tuning, udp workload tuning, and some other noteworthy parameters with the no utility. The article also addresses ARP cache tuning and how to monitor and tune ARP statistics. You will also look at name resolution and how you can easily increase performance by making small adjustments to resolve hostnames.

Monitoring network packets

In this section, you'll see an overview of tools available to help you monitor your network packets. These tools allow you to troubleshoot a performance problem quickly and capture data for historical trending and analysis.

[Part 1](#) of this series addressed some of the very basic flags, such as `-in`, that you typically use with `netstat`. Using `netstat`, you can also monitor more detailed information about the packets themselves. For example, the `-D` option shows the overall number of packets received, transmitted, and dropped in your communication subsystem. The results are sorted by device, driver, and protocol (see [Listing 1](#)).

Listing 1. netstat with the -D option

```
1
2                                     l488pp065_pub[/tmp] > netstat -D
3
4                                     Source          Ipkts          Opkts
5                                     Idrops        Odrops
6                                     -----
7                                     ent_dev1          71306227
8                                     337207          0          0
9
10                                    ent_dev0          203313084
11                                    82292          0          0
12
13                                     -----
14
15                                     Devices Total          274619311
16                                     419499          0          0
17
18                                     -----
19
20                                     -----
```

13	ent_dd1		71306227	
	337207	0	0	
14				
	ent_dd0		203313084	
15	82292	0	0	
16	-----			

17				
	Drivers Total		274619311	
18	419499	0	0	
19	-----			

20				
	ent_dmx1		70327758	N/A
21	978469	N/A		
22				
	ent_dmx0		202846759	N/A
23	466325	N/A		
24	-----			

25				
	Demuxer Total		273174517	
26	N/A	1444794	N/A	
27	-----			

28				
	IP		204276236	1063977
29	899839	828213		
30				
	IPv6		70588	70588
	0	6208		
31				
	TCP		714368	785630
	72	0		

32	UDP	202697468	319900
	202172157	0	
33			
34			
35	Protocols Total	407688072	2169507
	203072068	828213	
36			
37			
38	en_if1	70327759	
	337207	0	0
39			
	en_if0	202846780	
	82315	0	0
	lo_if0	780891	780890
	12	0	
	Net IF Total	273955430	
	1200412	12	0
	NFS/RPC Client	24	
	N/A	0	N/A
	NFS/RPC Server	0	
	N/A	0	N/A
	NFS Client	279	N/A
	0	N/A	

```
NFS Server          0          N/A
0      N/A

-----

-----

NFS/RPC Total      N/A
303    0    0

-----

-----

(Note: N/A -> Not Applicable)
```

Another useful flag is the `-s`, which shows detailed statistics for all protocols used, including packets sent, received and dropped. If you only want to view tcp, you can also use the `-p` flag (see [Listing 2](#)).

Listing 2. netstat with the -p option

```
1
2      1488pp065_pub[/tmp] > netstat -p tcptcp:
3
4      785684 packets sent
5
6      227657 data packets (13800898 bytes)
7
8      0 data packets (0 bytes) retransmitted
9
10     279285 ack-only packets (509 delayed)
11
12     0 URG only packets
13
14     0 window probe packets
```

8	69732 window update packets
9	418020 control packets
10	0 large sends
11	0 bytes sent using largesend
12	0 bytes is the biggest largesend
13	714418 packets received
14	360033 acks (for 14009902 bytes)
15	69728 duplicate acks
16	0 acks for unsent data
17	217241 packets (1660096 bytes) received in-sequence
18	72 completely duplicate packets (72 bytes)
19	0 old duplicate packets
20	0 packets with some dup. data (0 bytes duped)
21	69632 out-of-order packets (0 bytes)
22	0 packets (0 bytes) of data after window
23	0 window probes
24	69733 window update packets
25	0 packets received after close
26	

27	0 packets with bad hardware assisted checksum
28	
29	0 discarded for bad checksums
30	0 discarded for bad header offset fields
31	0 discarded because packet too short
32	0 discarded by listeners
33	0 discarded due to listener's queue full
34	5241 ack packet headers correctly predicted
35	75327 data packet headers correctly predicted
36	
37	69655 connection requests
38	69712 connection accepts
39	139364 connections established (including accepts)
40	139417 connections closed (including 12 drops)
41	
42	0 connections with ECN capability
43	0 times responded to ECN
44	2 embryonic connections dropped
45	429685 segments updated rtt (of 429463 attempts)

46	0 segments with congestion window reduced bit set
47	0 segments with congestion experienced bit set
48	0 resends due to path MTU discovery
49	1 path MTU discovery termination due to retransmits
50	3 retransmit timeouts
51	0 connections dropped by rexmit timeout
52	0 fast retransmits
53	0 when congestion window less than 4 segments
54	0 newreno retransmits
55	0 times avoided false fast retransmits
56	0 persist timeouts
57	0 connections dropped due to persist timeout
58	0 keepalive timeouts
59	0 keepalive probes sent
60	0 connections dropped by keepalive
61	0 times SACK blocks array is extended
62	0 times SACK holes array is extended
63	
64	

```

65          0 packets dropped due to memory allocation
           failure
66
           0 connections in timewait reused
67
           0 delayed ACKs for SYN
68
           0 delayed ACKs for FIN
69
           0 send_and_disconnects
70
           0 spliced connections
71
           0 spliced connections closed
           0 spliced connections reset
           0 spliced connections timeout
           0 spliced connections persist timeout
           0 spliced connections keepalive timeout
           0 TCP checksum offload disabled during
           retransmit
           0 Connections dropped due to bad ACKs

```

There are actually so many different ways of using netstat that the best place to start is by looking at the man page and go from there. Don't be afraid to run these commands, because they won't eat up disk space or affect performance. The tracing tools that are provided within AIX 7 are used to record detailed information about the packets. Use them with more caution. These tools are extremely helpful when trying to determine the root cause of network performance problems.

First, look at `iptrace` and `ipreport`. The `iptrace` command records all the packets received from the network interfaces. The `ipreport` command formats the data that is generated from `iptrace` into a readable trace report. Further, you can also use `ipfilter` to sort the output file created from `ipreport`. Try starting the trace and keep it going for one minute (see [Listing 3](#)).

Listing 3. Starting the trace

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

When you are done with the trace, you need to kill the process (see [Listing 4](#)).

Listing 4. Killing the process


```

11          IP:  ip_ttl=60, ip_sum=d60, ip_p = 6 (TCP)
12          TCP:  <source port=22(ssh), destination
13              port=54678 >
14          TCP:  th_seq=47587592, th_ack=3002348404
15          TCP:  th_off=8, flags<PUSH | ACK>
16          TCP:  th_win=65522, th_sum=0, th_urp=0
17          TCP:           nop
18          TCP:           nop
19          TCP:           timestamps TSV al: 0x4f486827
          TSEcho: 0x4c8da569
          TCP: 00000000  9fec0a46 c8dd1c9b 98ff0213
          87c714c0  |...F.....|
          TCP: 00000010  0ec081aa 7c76335f 0bfd0d8f
          63d0bf1a  |...lv3_...c...|
          TCP: 00000020  808359b4 13e1a29d 4dacdd51
          dad01053  |..Y.....M..Q...S|

```

Listing 5 shows the captured information about each packet, including packet size and IP address information. As you can imagine, the trace file can get very large quickly. The example file grew to 40MB in less than one minute! Be very careful when running these traces, because you will run out of disk space really fast if you don't have the disk bandwidth for these files.

You can also start the trace using the System Resource Controller (SRC). See [Listing 6](#).

Listing 6. Starting the trace using SRC

```
1
2
3
4
5
6
1488pp065_pub[/tmp] > startsrc -s iptrace -a "-i
en1 /home/testing/iptrace/iptracelog"0513-059 The
iptrace Subsystem
has been started. Subsystem PID is 12845270.
1488pp065_pub[/tmp] > stopsrc -s iptrace
0513-004 The Subsystem or Group, iptrace, is
currently inoperative.
```

What about tcpdump? tcpdump prints out headers of the packets, which are captured for each NIC. One important difference with tcpdump is that, unlike iptrace, it can look at only one network interface at a time. And, because iptrace examines the entire packet from the kernel space, the results can offer lots of dropped packets. With tcpdump, you can also limit the amount of data to be traced. Also, you do not need to use an ipreport type of command to format binary data, because tcpdump does the trace and the output. See [Listing 7](#) for an example.

Listing 7. Using tcpdump

```
1
2
1488pp065_pub[/tmp] > tcpdump > tcp.outtcpdump:
listening on
en0, link-type 1, capture size 96 bytes
```

tcpdump continues to capture packets until you hit Ctrl+C. If any packets were dropped due to a lack of buffer space, it reports that, too.

[Listing 8](#) shows what you see when you end the example trace and view the file.

Listing 8. End of the trace

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

```
28 packets received by filter0 packets dropped by
kernel

1488pp065_pub[/tmp] > cat tcp.out

06:00:21.003328 IP 1488pp065_pub.ssh >
172.29.131.16.54678:

P 47609416:47609464(48) ack 3002357700 win
65522 <nop,nop,timestamp 133014597

1 1284351989>

06:00:21.003387 IP 1488pp065_pub.ssh >
172.29.131.16.54678: P 48:208(160)

ack 1 win 65522 <nop,nop,timestamp 1330145971
1284351989>

06:00:21.028081 IP 172.29.131.16.54678 >
1488pp065_pub.ssh: . ack 208 win

32761 <nop,nop,timestamp 1284351989
1330145971>
```

15 06:00:21.238937 ARP, Request who-has
172.29.173.186 tell 172.29.133.221, length 46

16

17 06:00:21.239110 ARP, Request who-has
172.29.173.236 tell 172.29.129.59, length 46

18 06:00:21.325060 ARP, Request who-has
172.29.175.252 tell 172.29.129.58, length 46

19

20 06:00:21.464383 IP6 fe80::4464:ceff:fe65:4f0c >
ff02::1:ff41:34d0: ICMP6,

21 neighbor solicitation, who has
fe80::221:5eff:fe41:34d0, length

22

23 32

24 06:00:21.505281 ARP, Request who-has
172.29.175.60 tell 172.29.133.223, length 46

25 06:00:22.013530 ARP, Request who-has
172.29.174.66 tell 172.29.133.222, length 46

26

27 06:00:22.054164 ARP, Request who-has
172.29.173.237 tell 172.29.129.59, length 46

28 06:00:22.076819 ARP, Request who-has
172.29.122.25 tell 172.29.122.2, length 46

29

30 06:00:22.393898 IP 172.29.148.116.32852 >
239.255.255.253.svrloc: UDP, length 56

31 06:00:22.464355 IP6 fe80::4464:ceff:fe65:4f0c >
ff02::1:ff41:34d0: ICMP6, neighbor

32 solicitation, who has fe80::221:5eff:fe41:34d0,

33 length

```
34          32
35          06:00:22.935140 802.1d config
36          8000.00:16:60:f9:a8:00.8011 root
37          8000.00:16:60:f9:a8:00
38          pathcost 0 age 0 max 20 hello 2 fdelay 15
39          06:00:23.186380 ARP, Request who-has
40          172.29.122.26 tell 172.29.122.2, length 46
41          06:00:24.520770 ARP, Request who-has
42          172.29.175.60 tell 172.29.133.223, length 46
43          06:00:24.558139 ARP, Request who-has
44          172.29.175.252 tell 172.29.129.58, length 46
45          06:00:24.573524 ARP, Request who-has
46          172.29.175.5 tell 172.29.129.57, length 46
47          06:00:24.736838 IP 172.29.148.116.32853 >
48          239.255.255.253.svrloc: UDP, length 56
49          06:00:24.931436 802.1d config
50          8000.00:16:60:f9:a8:00.8011 root
51          8000.00:16:60:f9:a8:00
52          pathcost 0 age 0 max 20 hello 2 fdelay 15
53          06:00:25.029112 IP 172.29.133.222.netbios-dgm >
54          172.29.191.255.netbios-dgm: UDP, length
55          201
56          06:00:25.029965 IP 172.29.133.222.netbios-dgm >
57          172.29.191.255.netbios-dgm: UDP, length
58          201
```

06:00:25.030751 IP 172.29.133.222.netbios-dgm >

172.29.191.255.netbios-dgm: UDP, length
201

06:00:25.031674 IP 172.29.133.222.netbios-dgm >

172.29.191.255.netbios-dgm: UDP, length
201

06:00:25.032636 IP 172.29.133.222.netbios-dgm >

172.29.191.255.netbios-dgm: UDP, length
201

06:00:25.033647 IP 172.29.133.222.netbios-dgm >

172.29.191.255.netbios-dgm: UDP, length
201

06:00:25.033732 CDP v2, ttl: 180s, Device-ID
'Switch'[lcdp]

06:00:25.034738 IP 172.29.133.222.netbios-dgm >

172.29.191.255.netbios-dgm: UDP, length
201

06:00:25.035741 IP 172.29.133.222.netbios-dgm >

172.29.191.255.netbios-dgm: UDP, length
201


```
12          dgd_ping_time = 5
13          dgd_retry_time = 5
14          directed_broadcast = 0
15          fasttimo = 200
16          icmp6_errmsg_rate = 10
17          icmpaddressmask = 0
18          ie5_old_multicast_mapping = 0
19          ifsize = 256
20          ip6_defttl = 64
21          ip6_prune = 1
22          ip6forwarding = 0
23          ip6srcrouteforward = 1
24          ip_ifdelete_notify = 0
25          ip_nfrag = 200
26          ipforwarding = 0
27          ipfragttl = 2
28          ipignoreredirects = 0
29          ipqmaxlen = 100
30          ipsendredirects = 1
```

```
31 ipsrcrouteforward = 1
32 ipsrcrouterecv = 0
33 ipsrcrouteseend = 1
34 llsleep_timeout = 3
35 lo_perf = 1
36 lowthresh = 90
37 main_if6 = 0
38 main_site6 = 0
39 maxnip6q = 20
40 maxttl = 255
41 medthresh = 95
42 mpr_policy = 1
43 multi_homed = 1
44 nbc_limit = 262144
45 nbc_max_cache = 131072
46 nbc_min_cache = 1
47 nbc_ofile_hashsz = 12841
48 nbc_pseg = 0
49 nbc_pseg_limit = 524288
```

```
50 ndd_event_name = {all}
51 ndd_event_tracing = 0
52 ndp_mmaxtries = 3
53 ndp_umaxtries = 3
54 ndpqsize = 50
55 ndpt_down = 3
56 ndpt_keep = 120
57 ndpt_probe = 5
58 ndpt_reachable = 30
59 ndpt_retrans = 1
60 net_buf_size = {all}
61 net_buf_type = {all}
62 net_malloc_frag_mask = {0}
63 netm_page_promote = 1
64 nonlocsrcroute = 0
65 nstrpush = 8
66 passive_dgd = 0
67 pmtu_default_age = 10
68 pmtu_expire = 10
```



```
11 -----
12 nbc_limit          256K 256K 256K 0
13 8E-1 kbyte        D
14 thewall
15 -----
16 nbc_max_cache      128K 128K 128K 1
17 256M byte         D
18 nbc_min_cache
19 nbc_limit
20 -----
21 nbc_min_cache      1 1 1 1 128K
22 byte             D
23 nbc_max_cache
24 -----
25 nbc_ofile_hashsz   12841 12841 12841 1
26 999999 segment    D
-----
nbc_pseg            0 0 0 0 2G-1
segment            D
-----
```

```
-----  
nbc_pseg_limit      512K  512K  512K  0  
1M  kbyte          D  
  
-----  
-----  
ndd_event_name      {all} {all} {all} 0  
128  string        D  
  
... trimmed for clarity
```

There are many parameters here. The thewall defines the upper limit for network kernel buffers. Today, its size is defined at installation time, depending on the amount of RAM and the kernel type. For example, if you are running AIX 5.3 on a 64-bit kernel, the parameter is set at half the size of real memory.

Listing 11. Setting the size of the thewall parameter

```
1  
1488pp065_pub[/tmp] > no -algrep thewall  
2  
thewall = 1048576
```

[Part 1](#) discussed mbufs, but it's worth another mention here, because it relates to thewall. Remember that mbufs are used to store data in the kernel for both incoming and outgoing traffic. This is why determining the right amount of mbufs is extremely important. The value of the maxmbuf tunable limits

the amount of memory that the communication systems use. If the value is 0, thewall tunable is used, and it cannot be modified from its default. Changing this tunable is a way to lower the thewall limit. As the default, if maxmbuf is 0, this value is used regardless of what thewall uses. netstat -m is used to detect shortages of failures of network memory requests (see [Listing 12](#)).

Listing 12. netstat with the -m option

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
```

```
1488pp065_pub[/tmp] > netstat -m

Kernel malloc statistics:

***** CPU 0 *****

By size      inuse  calls failed  delayed  free
hiwat  freed

64          558 2087455   0      7  274
5240      0

128         5884 1901723   0     175  164
2620      0

256         5780 653578   0     295  2876
5240     500

512         7970 182051630  0     972
102  6550   0

1024        3159 1960612   0     794  49
2620      0

2048        1069 3462138   0     520  25
3930      0

4096        2056 2794     0     83   3
```

```

17          1310    0
18          8192          5    260    0    3    163
           327    0
19
           16384          256    413    0    62    0
20          163    0
21          32768          55    274    0    23    4
           81    0
22
           65536          117    175    0    76    0
23          81    0

           131072          4    5    0    0    102
           204    0

... other CPU stats trimmed

Streams mblk statistic failures:

0 high priority mblk failures

0 medium priority mblk failures

0 low priority mblk failures

```

In the example, there are no shortages (failures).

Although there are many parameters you can change with the no utility, most of them are better left alone. The most important parameters are ones that refer to TCP streaming workload tuning.

- `tcp_sendspace`—This controls how much buffer space in the kernel is used to buffer application data. You really want to bump this up from the default, because if its limit is reached, the sending application suspends data transfer until TCP sends the data to the buffer.
- `tcp_receivespace`—In addition to controlling the amount of buffer space to be consumed by receive buffers, AIX 7 also uses this value to determine the size to make its transmit window.
- `udp_sendspace`—With UDP, you can set this to no more than 65536, because IP has an upper limit of 65536 bytes per packet.
- `udp_receivespace`—This value should be greater than `udp_sendspace`, because it needs to handle as many simultaneous UDP packets per socket as it can. This parameter can easily be set to 10 times the value of `udp_sendspace`.

Now, let's make some changes. First, increase the size of `udp_sendspace` (see [Listing 13](#)).

Listing 13. Increasing the size of `udp_sendspace`

```

1                                     1488pp065_pub[/tmp] > no -p -o
2                                     udp_sendspace=65536Setting udp_sendspace to
                                     65536
3
                                     Setting udp_sendspace to 65536 in nextboot file
                                     Change to tunable udp_sendspace, will only be
                                     effective for future connections

```

Next, change `udp_receivespace` to the recommended configuration of 10 times `udp_sendspace`). See [Listing 14](#).

Listing 14. Changing `udp_receivespace`

```

1                                     1488pp065_pub[/tmp] > no -p -o
2                                     udp_receivespace=655360Setting udp_receivespace to

```


Other important workload parameters include `rfc1323` and `sb_max`.

The `rfc1323` tunable enables the TCP window scaling option, which allows TCP to use a larger window size. Turning it on enables the best TCP performance. The `sb_max` tunable sets an upper limit on the number of socket buffers queued to an individual socket, which controls the amount of buffer space consumed by buffers (queued to either a sender or received socket). This amount should usually be less than the wall and approximately 4 times the size of the largest value of the `tcp` or `udp` send and receive settings. For example, if your `udp_recvspace` is 655360, you can't go wrong if you double it to 1310720.

Now look at `tcp_nodelayack`. This tunable prompts TCP to send an immediate acknowledgement, rather than a delayed acknowledgement. While this can add more overhead in some environments, it can greatly improve network performance in others. If you change this parameter, but it does not improve performance, you can quickly change it back.

Next look at `ipqmalen`. This tunable controls the length of the IP input queue. If you see an overflow counter (through the use of `netstat -s`), setting a maximum length of this queue can help fix the overflow.

What about ARP? When many clients are connected to the system, you might want to tune the ARP cache. You can look at the statistics using `netstat` (see [Listing 17](#)).

Listing 17. Using `netstat` with `-p arp`

```
1                                     1488pp065_pub[/tmp] > netstat -p arparp:
2                                     12 packets sent
3                                     0 packets purged
```



```
tcp_sendspace 262144 tcp_recvspace 262144
rfc1323 1
```

You can change these options (by interface) by using SMIT, chdev, or ifconfig. Note that ifconfig will not update the Object Data Manager (ODM). Therefore, on a reboot, it will revert to the previous value. Because of that, you should use SMIT: the fastpath of **smit tcpip>further configuration>Network interfaces>Change/Show characteristics** of an interface (see [Figure 1](#)).

Figure 1. Using SMIT to change interface settings

You might wonder why the no parameters don't apply to some interfaces. Name resolution is another area that can impact performance. If you know how you want to resolve (using DNS or the hosts file), make sure name resolution is set up correctly in the /etc/netsvc.conf file. Look at a piece of the file in [Listing 20](#).

Listing 20. Piece of /etc/netsvc.conf file

```
1
```

```
2 # Example:
3 # aliases = nis, files
4 #
hosts=local,bind4
```

If you're using DNS, take out the local if you are not using a host's file at all, or you can leave it in if you are using it as a backup to DNS (but make it the second entry). Alternatively, take out the bind if you're not using DNS at all, because it will only slow down your performance by first attempting (if it is the first entry in the record) to resolve using a Name Server that doesn't exist.

Summary

This article discussed how to monitor network packets on the network. You used netstat and drilled down to the packet level using tracing tools, such as iptrace and tcpdump. Further, you learned how to tune your network using the no utility. Using this utility, you explored tcp and udp workload tuning while also learning some other noteworthy parameters. You made tuning changes and read about how you might want to tune certain settings. You also examined ARP cache tuning and saw how you could monitor and tune ARP statistics. You looked at ISNO and learned how you could tune specific no tunables by interface. You also looked at name resolution and how you could easily increase performance by making small adjustments in how to resolve hostnames.

Related topics

- [AIX Wiki](#): Get the nmon manual or downloads here.
- [Improving database performance with AIX concurrent I/O](#): Read this white paper for more

information on how to improve database performance.

- [Power Architecture: High-Performance Architecture with a History](#): Read this white paper.
- [nmon analyser -- A free tool to produce AIX performance reports](#) (Steven Atkins, developerWorks, April 2006): You can download nmon analyser from [here](#).
- [Operating System and Device Management](#) from IBM provides users and system administrators with complete information that can affect your selection of options when performing such tasks as backing up and restoring the system, managing physical and logical storage, and sizing appropriate paging space.
- [The AIX 7.1 Information Center](#) is your source for technical information about the AIX operating system.
- [The IBM AIX Version 6.1 Differences Guide](#) can be a useful resource for understanding changes in AIX 6.1.
- [AIX and UNIX](#): The AIX and UNIX developerWorks zone provides a wealth of information relating to all aspects of AIX systems administration and expanding your UNIX skills.
- [IBM trial software](#): Build your next development project with software for download directly from developerWorks.
- [Future Tech](#): Visit Future Tech's site to learn more about their latest offerings.

Posted - Mon, Jun 4, 2018 12:20 PM. This article has been viewed 6928 times.

Online URL: <http://kb.ictbanking.net/article.php?id=217>