

Page numbers followed by *f* indicate figures

## A

Accept syscall, 157–159, 157f, 158f  
 file table entry, 162  
 flow control, 162–163  
*inet\_accept()*, 159–161, 160f, 161f  
 inode and socket data structures linking, 161–162  
 VFS and socket data structures linking, 162

## Application interfaces for TCP/IP

client application, 27–29, 28f  
 option values  
   SO\_BROADCAST, 30  
   SO\_DEBUG, 29–30, 30f  
   SO\_DONTROUTE, 33  
   SO\_KEEPALIVE, 31, 32f  
   SO\_LINGER, 31–32  
   SO\_OOBINLINE, 32  
   SO\_RCVBUF, 33  
   SO\_RCVTIMEO, 33–34, 34f  
   SO\_REUSEADDR, 31  
   SO\_SNDBUF, 32–33  
   SO\_SNDTIMEO, 34–35, 35f  
 server application, 25–27, 26f  
 socket options, 29

ARP/RARP, 97–98, 97f, 98f

## B

Basic implementation, 1–2  
 BGP. *See* Border Gateway Protocol  
 Border Gateway Protocol (BGP), 90

## C

CBQ. *See* Class-based Queuing  
*CBQ\_dequeue()*, 623f, 624f  
*cbq\_dequeue()*, 627–629  
*cbq\_dequeue1()*, 629  
*cbq\_dequeue\_prio()*, 629–632

estimator, 625–626  
 general scheduler, 624  
 link-sharing scheduler, 625  
*from net/dev/core.c*, 626  
*qdisc\_restart()*, 626–627  
*qdisc\_run()*, 626

CBQ queuing discipline Icrash STEPS, 739

Class-based Queuing (CBQ), 622

Client side setup, 164f

client side operations, 164  
 connect, 164–167, 165f, 166f  
 flow control for connection request, 167–170, 168f, 169f  
*ip\_route\_connect()*, 167  
*tcp\_connect*, 174–176, 175f  
*tcp\_transmit\_skb()*, 176–178  
*tcp\_v4\_check\_established()*, 171–174  
*tcp\_v4\_connect()*, 167  
*tcp\_v4\_hash\_connect()*, 170–171

Compatibility framework

*FW\_ACCEPT* and *FW\_SKIP*, 647  
*fw\_in()*, 645–647  
*FW\_MASQUERADE*, 647  
*FW\_REDIRECT*, 647  
*FW\_REJECT*, 647

Connection queues, 733–735

Connection request handling, 151–154, 152f, 154f

accept queue processing, 155–156  
 flow control for handling a new connection request, 156  
 SYN queue processing, 155

Connection setup

  BIND, 124–125, 124f  
   *bind()*, 130  
   end of *fget()*, 131  
   end of *inet\_bind()*, 137  
   end of *sockfd\_lookup()*, 131

- Connection setup (*cont'd*)
  - end of tcp\_v4\_get\_port()*, 137
  - fget()*, 131
  - hash buckets for tcp Bind, 125
  - inet\_bind()*, 131–133
  - related data structures, 125
  - server side operations, 124
  - server side setup, 122–124, 123f
  - sockfd\_lookup()*, 130
  - sys\_bind()*, 130
  - tcp\_bhash*, 125–126
  - tcp\_bind\_bucket*, 129–130
  - tcp\_bind\_conflict()*, 135–136, 136f
  - tcp\_bind\_hashbucket*, 129
  - tcp\_ehash*, 125
  - tcp\_hashinfo*, 126–127, 127f
  - tcp\_listening\_hash*, 125
  - tcp\_v4\_get\_port()*, 133–135
- Core TCP processing, 444f
- D
- Data flow diagram, 284f–290f
- Data segments processing, 424–433
  - DSACK block and, 430, 430f, 431f
  - implementation, 425
  - tcp\_ofo\_queue()*, 436–441, 437f, 439f, 440f
  - tcp\_sack\_extend()*, 435–436
  - tcp\_sack\_maybe\_coalesce()*, 434–435
  - tcp\_sack\_new\_ofo\_skb()*, 433–434
  - tcp\_sack\_remove()*, 441–442
- Delay ack timer
  - ACK segments ending of, 344–345
  - quick ACK mode, 345
  - scheduling of, 344
  - tcp\_ack\_snd\_check()*, 346–347
  - \_tcp\_ack\_snd\_check()*, 345–346
  - tcp\_clear\_xmit\_timer()*, 352–353
  - tcp\_delack\_timer()*, 348–349
  - tcp\_reset\_xmit\_timer()*, 349–351
  - tcp\_send\_delayed\_ack()*, 347–348
  - tcp\_write\_timer()*, 351–352
- Duplicate/partial ACKs in loss state
  - tcp\_check\_sack\_reneging()*, 455–456
  - tcp\_try\_undo\_loss()*, 453–455
- Duplicate/partial ACKs in recovery state
  - tcp\_remove\_reno\_sacks()*, 450–451
  - tcp\_try\_undo\_partial()*, 451–452
- E
- Enqueue
  - cbq\_classify()*, 621
  - cbq\_enqueue()*, 620–621
- EWMA. *See* Exponential Weighted Moving Average
- Exponential Weighted Moving Average (EWMA), 625
- F
- FIB. *See* Forwarding Information Base
- FIB TABLE Icrash OUTPUT, 745–749, 746f, 747f, 748f
- Filters, 615–616
  - route filter implementation, 617f
    - route4\_change()*, 618–619
  - tc\_ctl\_tfilter()*, 613f, 611613
  - types of, 610
  - u32\_change()*, 615–616
  - u32 filter implementation, 614f
- Forwarding Information Base (FIB), 540
- Fragmentation and reassembly, 761
- I
- ICMP, 94f
  - ping*, 95–96, 95f, 96f
- Icrash output for route filter, 749–755, 750f, 751f, 752f, 753f, 754f
- Ikcd* source and patches, 724
- INET\_CREATE*, 111f
- I/O
  - read()*, 38, 38f
  - recv()*, 38, 38f
  - select()*, 39, 39f
  - send()*, 39, 39f
  - write()*, 38, 38f
- IP chains
  - definition of, 647
  - filtering with Ipchains, 648–649, 649f
  - Ipchain rules chains, 649
- IP tables
  - filtering packets, 664–668
  - filter rules, 657–658
  - ipt\_do\_table()*, 664–668
  - ipt\_match-iterate*, 668
  - registration of, 657
  - struct ipt\_entry*, 661–662
  - struct ipt\_entry\_match*, 662–663
  - struct ipt\_entry\_target*, 664
  - struct ipt\_standard\_target*, 664
  - struct ipt\_table*, 658
  - struct ipt\_table\_info*, 658–661, 660f
  - struct ipt\_tcp*, 663–664
- packet filtering
  - ip\_fw\_check()*, 653–655
  - ip\_rule\_match()*, 655
  - struct ip\_chain*, 649–650

- struct ip\_fw*, 651–652
- struct ip\_fwkernel*, 650–651
- struct ip\_reent*, 651
- table organization in, 652f
- IP forwarding, 761
- IP (Internet protocol)
  - IP header
    - checksum*, 89
    - dst addr.*, 90
    - flag.*, 89
    - frag offset*, 89
    - hlen*, 88
    - ID*, 89
    - prot.*, 89
    - src addr.*, 90
    - TOS*, 88
    - total len*, 88
    - TTL*, 89
    - ver.*, 88
- Ipv6, 761
- K
- Keepalive timer
  - activation of, 353–354
  - resetting of, 354
  - tcp\_keepalive\_timer()*, 354–356
- Kernel 2.6 description, 759
- Kernel flow, 214, 216f
- Kernel synchronization mechanism
  - atomic operations, 23
  - semaphore, 22
  - spin lock, 23–24, 24f
- Kernel version 2.4, 11–14, 13f, 14f
  - new system call addition, 16–17, 17f
  - system call on Linux, 14–16, 15f
- L
- Length reordering, 417–421, 418f
- Linux implementation of CBQ, 623f
- Linux process and thread
  - fork()*, 17–18, 18f
  - kernel threads, 19–21, 20f, 21f
  - thread, 18–19, 19f
- Linux traffic control
  - basic components of, 592, 592f
  - classes, 592
  - filters, 593
  - policing, 593
  - queuing discipline, 592
- Listen systemcall, 138f
  - accept queue is full, 147–150, 148f, 149f
  - connection request with complete three-way handshake, 151
  - connection request with pending three-way handshake, 150–151
  - END of *inet\_listen()*, 142
  - END of *tcp\_listen\_start()*, 142
  - established sockets linked in *tcp\_ehash* table, 150
  - inet\_listen()*, 139
  - listen flow, 142
  - max\_qlen\_log*, 140
  - qlen*, 140
  - qlen\_young*, 140
  - struct open\_request*, 142–147, 143f, 144f
  - SYN QUEUE, 140
  - syn\_table*, 140–141
  - sys\_listen()*, 138–139
  - tcp\_listen\_start()*, 139–142
- M
- Multicasting and broadcasting, 760
- N
- Nagle’s algorithm (RFC 896), 69–71, 69f, 70f, 71f
- Netfilter hook framework, 636–637
- Netfilter hooks on IP stack
  - hooks for incoming packets, 639–640
  - hooks for outgoing packets, 638–639, 638f
  - nf\_hook\_slow()*, 642–643
  - nf\_iterate()*, 643–644
  - processing of, 642
  - registration of, 640–642
  - struct nf\_hook\_ops*, 644
- Netlink data structure
  - nl\_table*, 755, 756f
  - rtnetlink\_link*, 755
- Netlink sockets
  - CLASS messages, 484
  - data structures
    - nl\_table*, 485–486, 486f
    - rtnetlink\_link*, 486–488
    - struct msghdr*, 489–490, 490f
    - struct nlmsghdr*, 488–489
  - FILTER messages, 484–485
  - flow diagram for TC command, 495–496, 496f
  - introduction of, 479–480
  - kernel netlink socket, creation of, 481–482
  - netlink packet format, 490
  - QDISC messages, 484
  - registration and initialization, 480–481
  - ROUTE messages, 484

- Netlink sockets (*cont'd*)
  - socket example
    - TC command flow in user space, 490–491, 491f
    - TC command in kernel space, 491–495
  - user netlink socket, creation of
    - ADDR messages, 484
    - LINK parameter messages, 483
- Net SoftIRQ, 672f
  - irq\_cpustat\_t*, 675
  - packet reception, 679–679, 680f, 682f
  - packet transmission, 686–695, 687f, 691f, 692f, 694f, 695f
  - processing of, 675–678, 682–686
  - reception, 672–675
  - registration for, 678–679
  - transmission, 672
  - variables for, 677
- New interface, addition of, 761
- O
- OOB data, sending of, 249–250
- Open Shortest Path First (OSPF), 90, 501
- OSPF. *See* Open Shortest Path First
- P
- Packet reception
  - DMA ring buffers and, 698
  - flow of, 698
  - process of, 698
  - reception ring buffer, 698–700, 700f
- Packet transmission, 701f
  - device initialization, 707
  - DMA receive ring buffers initialization, 709
  - DMA transit ring buffers initialization, 707–709
  - e100\_hardware\_send\_packet()*, 717
  - e100\_rx()*, 711–713
  - e100\_send\_packet()*, 713–717, 714f, 715f
  - e100tx\_interrupt()*, 720–721
  - flow of, with DMA, 702
  - implementation of reception, 704–705, 705f
  - Rx descriptors, 713
  - Rx DMA buffer initialization, 711
  - Rx interrupt and, 709–711, 710f
  - struct etrax\_dma\_descr*, 706–707
  - struct etrax\_eth\_descr*, 705–706
  - transmission ring buffer, 702, 703f
  - Tx DMA ring buffer initialization, 717, 718f, 719f
- Packet traversing
  - APR and neighboring framework, 212–213
  - INET protocol packet switcher, 223–224
  - IP layer, 206–207
  - kernel path for TCP, 209
    - IP layer, 211–212
    - IP layer routing, 210–211
    - netfilter hook, 212
    - packet scheduler and hard transmission, 213
    - socket layer, 210
    - TCP layer, 211
  - link layer, 207
  - packet reception, 219
  - packet scheduler, 207
  - from socket layer to device, 207–208, 208f
  - socket scheduler, 207
  - TCP layer, 206, 207
  - up the TCP/IP stack, 220f, 221f
    - from device to socket layer, 219, 220f, 221f
    - IP fragment handling, 223
    - IP layer, 215–216, 222–223
    - kernel path for TCP, 219–225
    - local input netfilter hook, 216
    - packet reception, 214
    - packet switcher, 222
    - pre-routing netfilter hook, 215
    - Rx SOFT IRQ, 214–215
    - Socket layer*, 225
    - SoftIRQ*, 219–220
    - TCP layer, 218–219, 224–225
- PFIFO\_FAST QDISC implementation, 593–596, 594f, 595f
- Processing TCP urgent pointer
  - tcp\_check\_urg()*, 422–424
- Protocol socket registration, 105f, 106f, 105107
- Q
- Qdisc. *See* Queuing Discipline
- Queuing discipline data structure
  - struct cbq\_class*, 599–601
  - struct Qdisc*, 596–597
  - struct Qdisc\_class\_ops*, 598–599
  - struct Qdisc\_ops*, 597–598
- Queuing Discipline (Qdisc), 591
- Queuing mechanism
  - lock\_sock()*, 265
  - \_lock\_sock()*, 265–266

- processing in tcp\_rcv\_established()*, 256–258
  - queue processing, 259–263, 260f, 261f
  - release\_sock()*, 266
  - \_release\_sock()*, 266–267
  - tcp data processing, 269f, 270f
    - cleanup\_rbuff()*, 268–270
    - data from receive buffer, 273
    - 1mss = n Bytes requested, 275
    - n Bytes requested, 276
    - n-X bytes requested, 275
    - one page requested, 276
    - paged buffer, 275–276, 275f
    - skb\_copy\_datagram\_iovec()*, 271–273, 272f
    - X bytes requested, 273–275, 274f
  - tcp\_data\_wait()*, 263–264, 264f
  - tcp\_prequeue()*, 258–259
  - tcp\_prequeue\_process()*, 264–265
- R**
- Receive side TCP memory management
    - general discussion, 305–308
    - \_skb\_queue\_purge()*, 317–319, 318f
    - tcp\_clamp\_window()*, 309–311
    - tcp\_collapse()*, 312–316, 314f, 316f
    - tcp\_collapse\_ofo\_queue()*, 311–312
    - tcp\_prune\_queue()*, 308–309
  - Retransmission and route, 732
  - RIP. *See* Routing Information Protocol
  - Routed packet, 214, 215f
  - Route filters, 743–745, 744f
  - Routines operating on *sk\_buff*
    - alloc\_skb()*, 190–191, 191f
    - skb\_pull()*, 195–196
    - skb\_push()*, 194–195, 195f
    - skb\_put()*, 192–194, 193f
    - skb\_reserve()*, 191–192
  - Routing
    - general description of, 501–503
    - multipathing, 505–509, 506f, 507f
      - change\_nexthops()*, 507–508, 508f
      - endfor\_nexthops()*, 508
      - FIB\_RES\_NH*, 508–509
    - netstat*, 90–91, 91f
    - policy-based routing, 504–505, 504f
    - record route options, 509–510
    - record routing, 510
    - routing cache data structures
      - struct dst\_entry*, 522–523
      - struct rtable*, 519–522
      - struct rt\_hash\_bucket*, 519
    - routing cache implementation, 517–519, 518f
    - routing protocols, 90
    - source routing
      - loose record routing, 511
      - SRR processing implementation, 511–517
      - strict source routing, 510–511
    - traceroute*, 92–93, 92f
  - Routing and IP Qos Icrash STEPS
    - steps for default queuing discipline, 735–738, 736f, 737f
  - Routing cache
    - cache timer, 530
    - dst\_destroy()*, 535–536
    - dst\_free()*, 534–535
    - \_dst\_free()*, 535
    - dst\_run()*, 536–537
    - fib\_create\_info()*, 557–558
    - FIB initialization, 562f
      - fib\_hash\_init()*, 562–563
      - fib\_rules\_init()*, 563
    - FIB overview, 540, 541f
    - FIB traversal flow diagram
      - fib\_lookup()*, 581–582
      - fn\_hash\_lookup()*, 584–585
      - \_in\_dev\_get()*, 577–578
      - inet\_select\_addr()*, 578–579
      - ip\_dev\_find()*, 576–577
      - ip\_route\_output()*, 563–564
      - ip\_route\_output\_key()*, 564–566, 565f
      - ip\_route\_output\_slow()*, 566–576
      - fn\_hash\_insert()*, 553–554, 553f, 558f
      - fn\_new\_zone()*, 554–555
    - for incoming packets, 529–530
    - inet\_rtm\_newroute()*, 550–551
    - inet\_rtm\_newrule()*, 559–560
    - interface down and *rt\_flush\_timer*, 537–538
    - link failure
      - dst\_link\_failure()*, 527
      - dst\_set\_expires()*, 528–529
      - ipv4\_link\_failure()*, 527–528
    - for local connections, 525–526
    - LPM algorithm and table lookup, 555–557
    - management of, 523–525
    - new entry addition, 549–550
    - route scopes
      - control flags, 581
      - types, 581
    - rt\_cache\_flush()*, 538–540

- Routing cache (*cont'd*)
  - rt\_may\_expire()*, 533–534
  - rt\_periodic\_timer*, 530–533
  - rules for, 583
  - \_sk\_dst\_check()*, 526–527
  - struct fib\_info*, 546–547
  - struct fib\_nh*, 547–548
  - struct fib\_node*, 544–545
  - struct fib\_rule*, 548–549
  - struct fib\_table*, 540–543
  - struct fn\_hash*, 543
  - struct fn\_zone*, 543–544
  - struct kern\_rta*, 552–553
  - struct rtmgs*, 551–552
- Routing Information Protocol (RIP), 90, 501
- Routing table, Linux kernel implementation, 517
- S
- Sack blocks, processing of
  - tcp\_sacktag\_write\_queue()*, 410–417, 411f, 413f
- Segmentation, functional level flow, 252f, 253f
- Segmentation with scatter-gather technique, 235–239, 236f
  - with scatter-gather support, 239, 239f
- Y bites and
  - can\_coalesce()*, 239–240
  - forced\_push()*, 241–242
  - skb\_entail()*, 248
  - tcp\_copy\_to\_page()*, 240–241
  - tcp\_mark\_push()*, 241
  - tcp\_minshall\_check()*, 245
  - tcp\_nagle\_check()*, 244–245
  - tcp\_push()*, 242–243
  - tcp\_push\_one()*, 247–248
  - \_tcp\_push\_pending\_frames()*, 243
  - tcp\_snd\_test()*, 243–244
  - tcp\_write\_xmit()*, 245–247
  - update\_send\_head()*, 247
- Send congestion window and *ssthresh*, 730–732, 731f
- Send socket buffer, 727–729, 728f
- Shutdown
  - kernel shutdown implementation
    - receive shutdown, 36–37, 37f
    - send shutdown, 36
    - values, needed for, 36
  - Sk\_buff* and
    - DMA-SKB\_FRAG\_STRUCT
    - DMA and *sk\_buff*, 188f
    - sk\_buff* and fragmentation, 190
    - sk\_buff* and IP fragmentation, 189f
  - Sk\_buff* Builds protocol headers
    - IP header, 197–198, 198f
    - link layer header, 198–199, 199f
    - tcp header, 196–197, 197f
  - Sk\_buff* Extracts protocol headers
    - datalink layer point, 199–200
    - IP layer header, 200
    - tcp layer header, 200–201, 201f
  - Sock, 112–118, 113f, 114f, 116f, 118f
  - Socket, touching of, 724–726, 725f
  - Socket buffer, 726–727, 727f
  - Sockets
    - SOCK\_ASYNC\_NOSPACE, 109
    - SOCK\_ASYNC\_WAITDATA, 109
    - SOCK\_NOSPACE, 110
    - states of BSD socket, 108
  - Sockets, kernel implementation of, 101–102, 102f, 107–108, 108f
  - Source code organization, 5–7, 6f, 7f
  - SRR processing implementation
    - ip\_forward\_options()*, 514–516
    - ip\_options\_compiled()*, 512
    - ip\_options\_rcv\_srr()*, 512–514
    - ip\_rt\_get\_source()*, 516–517
  - State processing
    - overview of, 446–448
  - Struct *skb\_shared\_info*, 186–187
  - Struct *sk\_buff*, 182–186, 183f
    - structure of, 182–186
  - Syn-ack timer
    - activation of, 356–357
    - cancellation of, 357
    - tcp\_synack\_timer()*, 357–361, 360f
  - Syn queues, 733–735
  - System-wide control parameters, 329–321
  - T
  - TC command in kernel space
    - netlink\_data\_ready()*, 494
    - netlink\_sendmsg()*, 492–493
    - netlink\_unicast()*, 493–494
    - rtnetlink\_rcv()*, 494
    - rtnetlink\_rcv\_msg()*, 494
    - rtnetlink\_rcv\_skb()*, 494
    - sock\_sendmsg()*, 492
    - sys\_sendmsg()*, 491–492
  - TCP
    - congestion control, 85–86
    - data flow
      - ACKing of data segments, 59–67, 60f, 61f, 63f, 64f, 65f, 66f

- delayed acknowledgment, 67–69, 67f, 68f
- header, 51f
  - acknowledgment number, 52
  - checksum, 53
  - header length, 52
  - port numbers, 52
  - sequence number, 52
  - TCP flags, 53
  - unused field, 53
  - urgent pointer, 53
  - window size, 53
- options, 54, 54f
  - mss option, 55, 55f
  - selective acknowledgment option, 57–58, 57f
  - timestamp option, 56
  - window-scaling option, 55–56, 56f
- performance and reliability
  - RTTD, 86
  - SACK/DSACK, 86–87
  - window scaling, 87
- sliding window protocol, 72–79, 74f, 75f, 76f, 77f, 78f
- timers
  - keepalive timer, 84
  - persistent timer, 83–85, 84f
  - retransmission timer, 88–83
  - TIME\_WAIT timer, 85
- TCP incoming segment processing, 378–379, 383
  - fast path enablement
    - processing of, 384–386
    - timing of, 382
  - prediction flags
    - building of, 383
    - important points, 383
  - prediction flags, building of, 378–380
  - processing of incoming ACK, 400–402
  - slow path enablement, 383
    - processing of, 386–387
  - tcp\_ack\_is\_dubious()*, 404
  - tcp\_ack\_update\_window()*, 406–407
  - tcp\_clean\_rtx\_queue*, 408–410
  - tcp\_cong\_avoid()*, 405–406
  - tcp\_data\_snd\_check()*, 397–398
  - \_tcp\_data\_snd\_check()*, 398
  - tcp\_event\_data\_recv()*, 390–391
  - tcp\_grow\_window()*, 392–393
  - \_tcp\_grow\_window()*, 393–394
  - tcp\_incr\_quickack()*, 391–392
  - tcp\_may\_update\_window()*, 407–408
  - tcp\_packets\_in\_flight()*, 403–404
  - tcp\_paws\_discard()*, 398–399
  - tcp\_receive\_window()*, 395
  - tcp\_replace\_ts\_recent()*, 387–389
  - tcp\_select\_window()*, 395–397
  - tcp\_sequence()*, 387
  - tcp\_space()*, 397
  - window calculation, 394–395
- TCP/IP stack overview
  - INET socket in, 3
  - kernel control paths and, 7–11
  - kernel networking source tree, 9f
  - kernel source tree, 8f
  - packet moving down protocol stack, 3, 4f
  - packet moving up protocol stack, 5
  - packet reception, 11f
  - sk\_buff*, 2f, 3
- TCP retransmit timer
  - resetting and cancellation, 327–329
  - setting of, 327
  - skb\_cloned()*, 336
  - tcp\_enter\_loss()*, 329–332
  - tcp\_retransit\_skb()*, 333–334
  - tcp\_retrans\_try\_collapse()*, 334–336
- TCP segmentation unit, 729–730, 730f
  - functioning of, 232–233, 233f, 238f
  - segmentation without scatter-gather support, 234
- TCP states
  - categories of, 40
  - complete life cycle, 42f
  - connection closure, 40
  - connection initiation, 40
  - default processing, 456–459
  - established connection, 40
  - four-way connection closure, 43f
  - non-open states when acked beyond
    - tcp\_add\_reno\_sack()*, 472–473
    - TCP\_CA\_CWR*, 468–470
    - TCP\_CA\_Disorder*, 470–471
    - TCP\_CA\_Loss*, 467–468
    - TCP\_CA\_Recovery()*, 471–472
    - tcp\_check\_reno\_reordering()*, 473
    - tcp\_mark\_head\_lost()*, 475–477
    - tcp\_may\_undo()*, 473–474
    - tcp\_packet\_delayed()*, 474–475
    - tcp\_sync\_left\_out()*, 477
    - tcp\_try\_undo\_dsack()*, 471
    - tcp\_undo\_cwr()*, 475
  - partial close, 45–47, 46f
  - TCP\_CA\_CWR*, 449
  - tcp\_head\_timeout()*, 460–461
  - tcp\_packet\_delayed()*, 466–467

- TCP states (*cont'd*)
  - tcp\_time\_to\_recover()*, 459–460
  - tcp\_try\_to\_open()*, 461–462
  - tcp\_update\_scoreboard()*, 462–464
  - tcp\_xmit\_retransit\_queue()*, 464–466
  - three-way handshake, 40f, 41f
  - TIME\_WAIT, 44–45
  - undoing from TCP\_CA\_CWR, 449
- TCP throughput, maximizing of
  - bandwidth*, 79
  - congestion window, 80f, 81f
  - rtt* (round trip time), 79
- TC user program
  - cbq\_init()*, 604
  - commands for hierarchy creation
    - cbq\_change\_class()*, 607–610
    - tc\_ctl\_tclass()*, 606–607
    - dev\_graft\_qdisc()*, 605
    - qdisc\_create()*, 602–604
    - qdisc\_graft()*, 604–605
    - tc\_modify\_qdisc()*, 601–602
- Timers in Linux
  - detach\_timer()*, 325
  - mod\_timer()*, 324–325
  - time routines execution, 326
- Timers in Linux
  - del\_timer()*, 325–326
- Time\_wait timer
  - activation of, 361–362
  - non-recycle mode, 363–364, 365f
  - recycle mode, 365–367, 366f
  - tcp\_time\_wait()*, 362
  - tcp\_twcal\_tick()*, 370–374, 371f, 373f
  - \_tcp\_tw-hashdance()*, 374–375
  - tcp\_twkill()*, 367–370, 368f
  - tcp\_tw\_schedule()*, 362–363
- Transit side TCP memory management,
  - 291–294, 293f
  - alloc\_skb()*, 296–297
  - select\_size()*, 294–295
  - skb\_charge()*, 298
  - sock\_wfree()*, 300–301
  - tcp\_alloc\_page()*, 297–298
  - tcp\_alloc\_pskb()*, 295–296
  - tcp\_free\_skb()*, 300
  - tcp\_mem\_reclaim()*, 302
  - \_tcp\_mem\_reclaim()*, 302–303
  - tcp\_mem\_schedule()*, 298–300
  - tcp\_write\_space()*, 301–302
  - wait\_or\_tcp\_memory()*, 303–305, 304f
- U
- UDP, 760
- U32 filters, 739–743, 740f, 742f
- Urgent byte processing, 277f
  - byte read as OOB data, 277–278
  - reading as inline data, 280–284, 282f, 283f, 284f
  - tcp\_recv\_urg()*, 278–280, 279f
- V
- VFS and socket, 103–105, 103f, 104f
- Z
- Zero window probe timer
  - cancellation of, 337
  - function of, 338–339, 338f
  - installation of, 337
  - tcp\_ack\_probe()*, 338
  - tcp\_probe\_timer()*, 339
  - tcp\_send\_probe0()*, 339
  - tcp\_write\_wakeup()*, 339–342