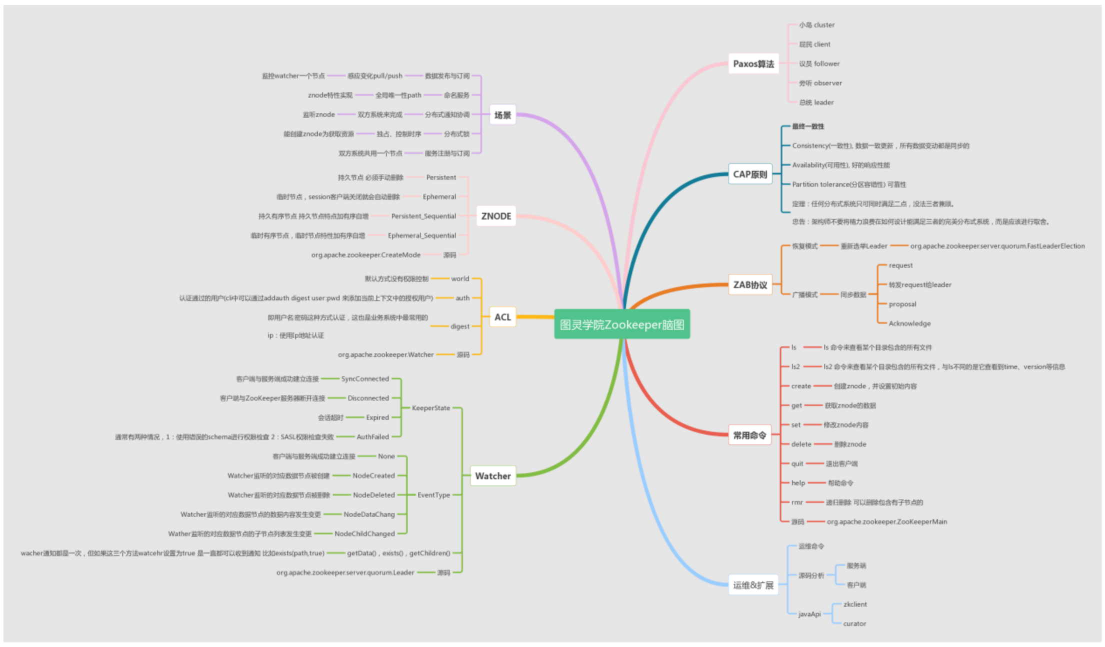
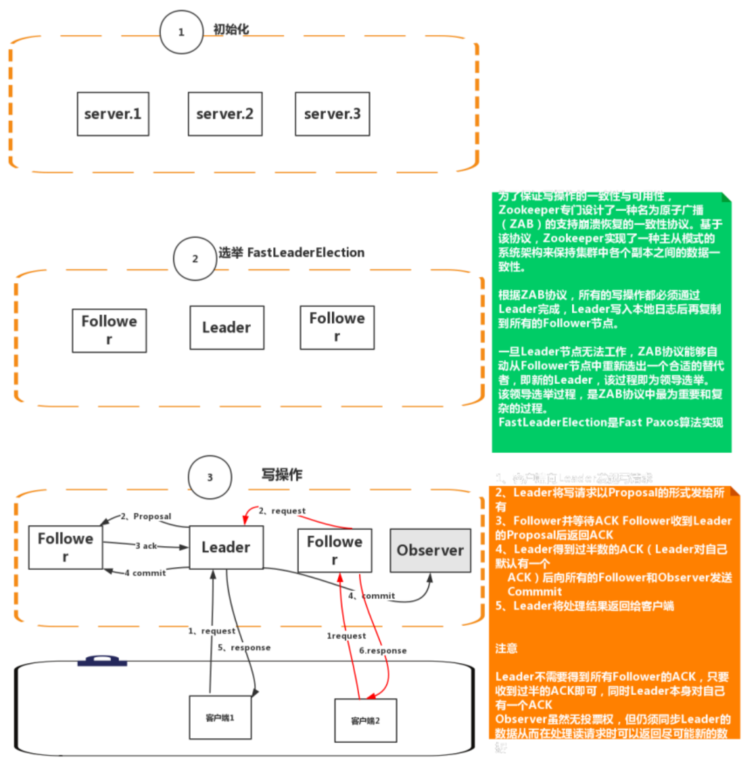

图灵学院Zookeeper脑图
场景
数据发布与订阅 — 感应变化pull/push — 监控watcher一个节点
命名服务 — 全局唯一性path — znode特性实现
分布式通知协调 — 双方系统来完成 — 监听znode
分布式锁 — 独占、控制时序 — 能创建znode为获取资源
服务注册与订阅 — 双方系统共用一个节点
ZNODE
Persistent — 持久节点 必须手动删除
Ephemeral — 临时节点，session客户端关闭就会自动删除
Persistent_Sequential — 持久有序节点 持久节点特点加有序自增
Ephemeral_Sequential — 临时有序节点，临时节点特性加有序自增
源码 — org.apache.zookeeper.CreateMode
ACL
world — 默认方式没有权限控制
auth — 认证通过的用户(cli中可以通过addauth digest user:pwd 来添加当前上下文中的授权用户)
digest — 即用户名:密码这种方式认证，这也是业务系统中最常用的
ip：使用Ip地址认证
源码 — org.apache.zookeeper.Watcher
Watcher
KeeperState
SyncConnected — 客户端与服务端成功建立连接
Disconnected — 客户端与ZooKeeper服务器断开连接
Expired — 会话超时
AuthFailed — 通常有两种情况，1：使用错误的schema进行权限检查 2：SASL权限检查失败
EventType
None — 客户端与服务端成功建立连接
NodeCreated — Watcher监听的对应数据节点被创建
NodeDeleted — Watcher监听的对应数据节点被删除
NodeDataChang — Watcher监听的对应数据节点的数据内容发生变更
NodeChildChanged — Wather监听的对应数据节点的子节点列表发生变更
getData()，exists()，getChildren() — wacher通知都是一次，但如果这三个方法watcehr设置为true 是一直都可以收到通知 比如exists(path,true)
源码 — org.apache.zookeeper.server.quorum.Leader
Paxos算法
小岛 cluster
屁民 client
议员 follower
旁听 observer
总统 leader
CAP原则
最终一致性
Consistency(一致性), 数据一致更新，所有数据变动都是同步的
Availability(可用性), 好的响应性能
Partition tolerance(分区容错性) 可靠性
定理：任何分布式系统只可同时满足二点，没法三者兼顾。
忠告：架构师不要将精力浪费在如何设计能满足三者的完美分布式系统，而是应该进行取舍。
ZAB协议
恢复模式 — 重新选举Leader — org.apache.zookeeper.server.quorum.FastLeaderElection
广播模式 — 同步数据
request
转发request给leader
proposal
Acknowledge
常用命令
ls — ls 命令来查看某个目录包含的所有文件
ls2 — ls2 命令来查看某个目录包含的所有文件，与ls不同的是它查看到time、version等信息
create — 创建znode，并设置初始内容
get — 获取znode的数据
set — 修改znode内容
delete — 删除znode
quit — 退出客户端
help — 帮助命令
rmr — 递归删除 可以删除包含有子节点的
源码 — org.apache.zookeeper.ZooKeeperMain
运维&扩展
运维命令
源码分析
服务端
客户端
javaApi
zkclient
curator



为了保证写操作的一致性与可用性，Zookeeper专门设计了一种名为原子广播（ZAB）的支持崩溃恢复的一致性协议。基于该协议，Zookeeper实现了一种主从模式的系统架构来保持集群中各个副本之间的数据一致性。

根据ZAB协议，所有的写操作都必须通过Leader完成，Leader写入本地日志后再复制到所有的Follower节点。

一旦Leader节点无法工作，ZAB协议能够自动从Follower节点中重新选出一个合适的替代者，即新的Leader，该过程即为领导选举。该领导选举过程，是ZAB协议中最为重要和复杂的过程。
FastLeaderElection是Fast Paxos算法实现

1、客户端向Leader发起写请求
2、Leader将写请求以Proposal的形式发给所有
3、Follower并等待ACK Follower收到Leader的Proposal后返回ACK
4、Leader得到过半数的ACK（Leader对自己默认有一个
   ACK）后向所有的Follower和Observer发送Commmit
5、Leader将处理结果返回给客户端

注意

Leader不需要得到所有Follower的ACK，只要收到过半的ACK即可，同时Leader本身对自己有一个ACK
Observer虽然无投票权，但仍须同步Leader的数据从而在处理读请求时可以返回尽可能新的数据

服务启动：

```
public static void main(String[] args) {
    QuorumPeerMain main = new QuorumPeerMain();
    try {
        main.initializeAndRun(args);//点这看
    } catch (IllegalArgumentException e) {
        LOG.error("Invalid arguments, exiting abnormally", e);
        LOG.info(USAGE);
        System.err.println(USAGE);
        System.exit(2);
    } catch (ConfigException e) {
        LOG.error("Invalid config, exiting abnormally", e);
        System.err.println("Invalid config, exiting abnormally");
        System.exit(2);
    } catch (Exception e) {
        LOG.error("Unexpected exception, exiting abnormally", e);
        System.exit(1);
    }
    LOG.info("Exiting normally");
    System.exit(0);
}
```

org.apache.zookeeper.server.quorum.QuorumPeerMain#initializeAndRun

```
protected void initializeAndRun(String[] args)
    throws ConfigException, IOException
{   //读取 zoo.cfg 配置参数
    QuorumPeerConfig config = new QuorumPeerConfig();
    if (args.length == 1) {
        config.parse(args[0]);
    }

    // Start and schedule the the purge task

    //启动日志清除任务
    DatadirCleanupManager purgeMgr = new DatadirCleanupManager(config
            .getDataDir(), config.getDataLogDir(), config
            .getSnapRetainCount(), config.getPurgeInterval());
    purgeMgr.start();

    if (args.length == 1 && config.servers.size() > 0) {
        runFromConfig(config); //读取到的配置进行搞事 xxoo 哈哈
    } else {
        LOG.warn("Either no config or no quorum defined in config, running "
                + " in standalone mode");
        // there is only server in the quorum -- run as standalone
        ZooKeeperServerMain.main(args);
    }
}
```

org.apache.zookeeper.server.quorum.QuorumPeerMain#runFromConfig

```
public void runFromConfig(QuorumPeerConfig config) throws IOException {
  try {
      ManagedUtil.registerLog4jMBeans();
  } catch (JMException e) {
      LOG.warn("Unable to register log4j JMX control", e);
  }

  LOG.info("Starting quorum peer");
  try {
      ServerCnxnFactory cnxnFactory = ServerCnxnFactory.createFactory();
      cnxnFactory.configure(config.getClientPortAddress(),
                          config.getMaxClientCnxns());//创建服务端的 Socket 实
列

      quorumPeer = new QuorumPeer();//confg 读取到的 zoo.cfg 赋值
      quorumPeer.setClientPortAddress(config.getClientPortAddress());
      quorumPeer.setTxnFactory(new FileTxnSnapLog(
                  new File(config.getDataLogDir()),
                  new File(config.getDataDir())));
      quorumPeer.setQuorumPeers(config.getServers());
      quorumPeer.setElectionType(config.getElectionAlg());
      quorumPeer.setMyid(config.getServerId());
      quorumPeer.setTickTime(config.getTickTime());
      quorumPeer.setMinSessionTimeout(config.getMinSessionTimeout());
      quorumPeer.setMaxSessionTimeout(config.getMaxSessionTimeout());
      quorumPeer.setInitLimit(config.getInitLimit());
      quorumPeer.setSyncLimit(config.getSyncLimit());
      quorumPeer.setQuorumVerifier(config.getQuorumVerifier());
      quorumPeer.setCnxnFactory(cnxnFactory);
      quorumPeer.setZKDatabase(new ZKDatabase(quorumPeer.getTxnFactory()));
      quorumPeer.setLearnerType(config.getPeerType());
      quorumPeer.setSyncEnabled(config.getSyncEnabled());
      quorumPeer.setQuorumListenOnAllIPs(config.getQuorumListenOnAllIPs());

      quorumPeer.start(); //调用 start 方法 注意这不是调用线程的 start 方法
      quorumPeer.join();
  } catch (InterruptedException e) {
      // warn, but generally this is ok
      LOG.warn("Quorum Peer interrupted", e);
  }
}
```

org.apache.zookeeper.server.quorum.QuorumPeer#start

```
@Override
public synchronized void start() {
    loadDataBase();//先从内存中恢复数据写到文件中
    cnxnFactory.start();   //启动服务器端 Socket 实现
    startLeaderElection();//开始选举
    super.start();//这才真正调用线程的 start 方法也就会执行 run 方法
}
```

org.apache.zookeeper.server.NIOServerCnxnFactory#run 服务端建立链接

```

public void run() {
    while (!ss.socket().isClosed()) {
        try {
            selector.select(1000);
            Set<SelectionKey> selected;
            synchronized (this) {
                selected = selector.selectedKeys();
            }
            ArrayList<SelectionKey> selectedList = new ArrayList<SelectionKey>(
                    selected);
            Collections.shuffle(selectedList); //乱序
            for (SelectionKey k : selectedList) {
                if ((k.readyOps() & SelectionKey.OP_ACCEPT) != 0) {
                    SocketChannel sc = ((ServerSocketChannel) k
                            .channel()).accept();
                    InetAddress ia = sc.socket().getInetAddress();
                    int cnxncount = getClientCnxnCount(ia);

                      //调用 zoo.cfg 配置的客户端连接数是否超过了
                    if (maxClientCnxns > 0 && cnxncount >= maxClientCnxns){
                        LOG.warn("Too many connections from " + ia
                                 + " - max is " + maxClientCnxns );
                        sc.close();
                    } else {
                        LOG.info("Accepted socket connection from "
                                 + sc.socket().getRemoteSocketAddress());
                        sc.configureBlocking(false);

                      //监听 read 事件
                        SelectionKey sk = sc.register(selector,
                                SelectionKey.OP_READ);

                      //创建内部
                        NIOServerCnxn cnxn = createConnection(sc, sk);
                        sk.attach(cnxn);
                        addCnxn(cnxn);
                    }
                } else if ((k.readyOps() & (SelectionKey.OP_READ |
SelectionKey.OP_WRITE)) != 0) {//处理读和写事件操作
                    NIOServerCnxn c = (NIOServerCnxn) k.attachment();
                    c.doIO(k);//不建议跟下去了
                } else {
                    if (LOG.isDebugEnabled()) {
                        LOG.debug("Unexpected ops in select "
                                  + k.readyOps());
                    }
                }
            }

              //清除 下次之需
            selected.clear();
        } catch (RuntimeException e) {
            LOG.warn("Ignoring unexpected runtime exception", e);
        } catch (Exception e) {
            LOG.warn("Ignoring exception", e);
        }
    }
    closeAll();
    LOG.info("NIOServerCnxn factory exited run method");
}
```

org.apache.zookeeper.server.quorum.QuorumPeer#startLeaderElection 选举开始

```
synchronized public void startLeaderElection() {
   try {
      currentVote = new Vote(myid, getLastLoggedZxid(), getCurrentEpoch());

        //投票给自己
   } catch(IOException e) {
      RuntimeException re = new RuntimeException(e.getMessage());
      re.setStackTrace(e.getStackTrace());
      throw re;
   }

      //从配置中拿自己的选举地址
    for (QuorumServer p : getView().values()) {
        if (p.id == myid) {
            myQuorumAddr = p.addr;
            break;
        }
    }
    if (myQuorumAddr == null) {
        throw new RuntimeException("My id " + myid + " not in the peer list");
    }
    if (electionType == 0) {
        try {
            udpSocket = new DatagramSocket(myQuorumAddr.getPort());
            responder = new ResponderThread();
            responder.start();
        } catch (SocketException e) {
            throw new RuntimeException(e);
        }
    }
    this.electionAlg = createElectionAlgorithm(electionType); //这是选举的开始
}
```

org.apache.zookeeper.server.quorum.FastLeaderElection#starter 选举初始化

```
private void starter(QuorumPeer self, QuorumCnxManager manager) {
    this.self = self;
    proposedLeader = -1;
    proposedZxid = -1;

    sendqueue = new LinkedBlockingQueue<ToSend>();
    recvqueue = new LinkedBlockingQueue<Notification>();
    this.messenger = new Messenger(manager);
}
```

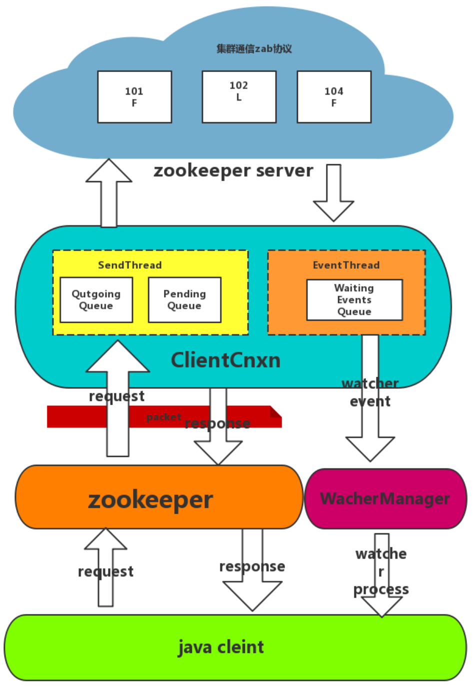
org.apache.zookeeper.server.quorum.QuorumPeer#run 选举开始

这就不贴代码了

其次可以看看

FastLeaderElection 中的 lookForLeader 方法 在这个 run 方法中会调用它 产生 leader 和 follower

架构师之家 www.jiagouedu.com 图灵技术交流网站 http://www.jiagouedu.com



outgoingQueue需要发送服务端
pendingQueue 已经发送等待服务响应的
waitingeventsQueue临时存放需要被触发的对象

packet请求type   ZooDefs>OpCode

客户端：

```
public ZooKeeper(String connectString, int sessionTimeout, Watcher watcher,
        boolean canBeReadOnly)
    throws IOException
{
    LOG.info("Initiating client connection, connectString=" + connectString
            + " sessionTimeout=" + sessionTimeout + " watcher=" + watcher);

    watchManager.defaultWatcher = watcher;

    ConnectStringParser connectStringParser = new ConnectStringParser(
            connectString);
    HostProvider hostProvider = new StaticHostProvider(
            connectStringParser.getServerAddresses());//拿到 ip 端口号
    cnxn = new ClientCnxn(connectStringParser.getChrootPath(),
            hostProvider, sessionTimeout, this, watchManager,
            getClientCnxnSocket(), canBeReadOnly);//创建 ClientCnxn 对象
    cnxn.start();//非 thread 线程启动
}
```

org.apache.zookeeper.ClientCnxn#ClientCnxn 初始化 启动了两个线程 send 和 event

```
public ClientCnxn(String chrootPath, HostProvider hostProvider, int
sessionTimeout, ZooKeeper zooKeeper,
        ClientWatchManager watcher, ClientCnxnSocket clientCnxnSocket,
        long sessionId, byte[] sessionPasswd, boolean canBeReadOnly) {
    this.zooKeeper = zooKeeper;
    this.watcher = watcher;
    this.sessionId = sessionId;
    this.sessionPasswd = sessionPasswd;
    this.sessionTimeout = sessionTimeout;
    this.hostProvider = hostProvider;
    this.chrootPath = chrootPath;

    connectTimeout = sessionTimeout / hostProvider.size();
    readTimeout = sessionTimeout * 2 / 3;
    readOnly = canBeReadOnly;

    sendThread = new SendThread(clientCnxnSocket);
    eventThread = new EventThread();

}
```

org.apache.zookeeper.ClientCnxn#start 连个线程 start>run 方法

```
public void start() {
    sendThread.start();
    eventThread.start();
}
```

org.apache.zookeeper.ClientCnxn.SendThread#run

```
@Override
public void run() {
    clientCnxnSocket.introduce(this,sessionId);
    clientCnxnSocket.updateNow();
    clientCnxnSocket.updateLastSendAndHeard(); //客户端和服务端链接的 socket 更新
    int to;
    long lastPingRwServer = System.currentTimeMillis();
    final int MAX_SEND_PING_INTERVAL = 10000; //10 seconds
    while (state.isAlive()) {
        try {
            if (!clientCnxnSocket.isConnected()) {
                if(!isFirstConnect){
                    try {
                        Thread.sleep(r.nextInt(1000));
                    } catch (InterruptedException e) {
                        LOG.warn("Unexpected exception", e);
                    }
                }
                // don't re-establish connection if we are closing
                if (closing || !state.isAlive()) {
                    break;
                }
                startConnect();
                clientCnxnSocket.updateLastSendAndHeard();
            }

            if (state.isConnected()) {
                // determine whether we need to send an AuthFailed event.
                if (zooKeeperSaslClient != null) {
                    boolean sendAuthEvent = false;
                    if (zooKeeperSaslClient.getSaslState() ==
ZooKeeperSaslClient.SaslState.INITIAL) {
                        try {
                            zooKeeperSaslClient.initialize(ClientCnxn.this);
                        } catch (SaslException e) {
                           LOG.error("SASL authentication with Zookeeper Quorum member failed:
" + e);
                            state = States.AUTH_FAILED;
                            sendAuthEvent = true;
                        }
                    }
                    KeeperState authState = zooKeeperSaslClient.getKeeperState();
                    if (authState != null) {
                        if (authState == KeeperState.AuthFailed) {
                            // An authentication error occurred during authentication with the
Zookeeper Server.
                            state = States.AUTH_FAILED;
                            sendAuthEvent = true;
                        } else {
                            if (authState == KeeperState.SaslAuthenticated) {
                                sendAuthEvent = true;
                            }
                        }
                    }

                    if (sendAuthEvent == true) {
                        eventThread.queueEvent(new WatchedEvent(
                              Watcher.Event.EventType.None,
                              authState,null));
                    }
                }
                to = readTimeout - clientCnxnSocket.getIdleRecv();
            } else {
                to = connectTimeout - clientCnxnSocket.getIdleRecv();
            }

            if (to <= 0) {
                String warnInfo;
                warnInfo = "Client session timed out, have not heard from server in "
                    + clientCnxnSocket.getIdleRecv()
                    + "ms"
                    + " for sessionid 0x"
                    + Long.toHexString(sessionId);
                LOG.warn(warnInfo);
                throw new SessionTimeoutException(warnInfo);
            }
            if (state.isConnected()) {
               //1000(1 second) is to prevent race condition missing to send the second ping
               //also make sure not to send too many pings when readTimeout is small
                int timeToNextPing = readTimeout / 2 - clientCnxnSocket.getIdleSend() -
                      ((clientCnxnSocket.getIdleSend() > 1000) ? 1000 : 0);
                //send a ping request either time is due or no packet sent out within
```

```
MAX_SEND_PING_INTERVAL
                if (timeToNextPing <= 0 || clientCnxnSocket.getIdleSend() >
MAX_SEND_PING_INTERVAL) {
                    sendPing();//发送心跳
                    clientCnxnSocket.updateLastSend();
                } else {
                    if (timeToNextPing < to) {
                        to = timeToNextPing;
                    }
                }
            }

            // If we are in read-only mode, seek for read/write server
            if (state == States.CONNECTEDREADONLY) {
                long now = System.currentTimeMillis();
                int idlePingRwServer = (int) (now - lastPingRwServer);
                if (idlePingRwServer >= pingRwTimeout) {
                    lastPingRwServer = now;
                    idlePingRwServer = 0;
                    pingRwTimeout =
                        Math.min(2*pingRwTimeout, maxPingRwTimeout);
                    pingRwServer();
                }
                to = Math.min(to, pingRwTimeout - idlePingRwServer);
            }

            clientCnxnSocket.doTransport(to, pendingQueue, outgoingQueue, ClientCnxn.this);

                //这个方法比较长 重点看这
        } catch (Throwable e) {
            if (closing) {
                if (LOG.isDebugEnabled()) {
                    // closing so this is expected
                    LOG.debug("An exception was thrown while closing send thread for session
0x"
                            + Long.toHexString(getSessionId())
                            + " : " + e.getMessage());
                }
                break;
            } else {
                // this is ugly, you have a better way speak up
                if (e instanceof SessionExpiredException) {
                    LOG.info(e.getMessage() + ", closing socket connection");
                } else if (e instanceof SessionTimeoutException) {
                    LOG.info(e.getMessage() + RETRY_CONN_MSG);
                } else if (e instanceof EndOfStreamException) {
                    LOG.info(e.getMessage() + RETRY_CONN_MSG);
                } else if (e instanceof RWServerFoundException) {
                    LOG.info(e.getMessage());
                } else {
                    LOG.warn(
                            "Session 0x"
                                    + Long.toHexString(getSessionId())
                                    + " for server "
                                    + clientCnxnSocket.getRemoteSocketAddress()
                                    + ", unexpected error"
                                    + RETRY_CONN_MSG, e);
                }
                cleanup();
                if (state.isAlive()) {
                    eventThread.queueEvent(new WatchedEvent(
                            Event.EventType.None,
                            Event.KeeperState.Disconnected,
                            null));
                }
                clientCnxnSocket.updateNow();
                clientCnxnSocket.updateLastSendAndHeard();
            }
        }
    }
    cleanup();
    clientCnxnSocket.close();
    if (state.isAlive()) {
        eventThread.queueEvent(new WatchedEvent(Event.EventType.None,
                Event.KeeperState.Disconnected, null));
    }
    ZooTrace.logTraceMessage(LOG, ZooTrace.getTextTraceLevel(),
            "SendThread exited loop for session: 0x"
                    + Long.toHexString(getSessionId()));
}
```

org.apache.zookeeper.ClientCnxnSocketNIO#doTransport 真正干事的

```
@Override
void doTransport(int waitTimeOut, List<Packet> pendingQueue, LinkedList<Packet> 
outgoingQueue,
                 ClientCnxn cnxn)
        throws IOException, InterruptedException {
    selector.select(waitTimeOut);
    Set<SelectionKey> selected;
    synchronized (this) {
        selected = selector.selectedKeys();
    }
    // Everything below and until we get back to the select is
    // non blocking, so time is effectively a constant. That is
    // Why we just have to do this once, here
    updateNow();
    for (SelectionKey k : selected) {
        SocketChannel sc = ((SocketChannel) k.channel());
        if ((k.readyOps() & SelectionKey.OP_CONNECT) != 0) {
            if (sc.finishConnect()) {
                updateLastSendAndHeard();
                sendThread.primeConnection();
            }
        } else if ((k.readyOps() & (SelectionKey.OP_READ | 
SelectionKey.OP_WRITE)) != 0) {
            doIO(pendingQueue, outgoingQueue, cnxn);//这是处理客户端往服务端发送
的数据 链接之后会处理读和写操作 这不往下跟代码了        }
    }
    if (sendThread.getZkState().isConnected()) {
        synchronized(outgoingQueue) {
            if (findSendablePacket(outgoingQueue,

cnxn.sendThread.clientTunneledAuthenticationInProgress()) != null) {
                enableWrite();
            }
        }
    }
    selected.clear();
}
```

如果是回调函数怎么处理了

org.apache.zookeeper.ClientCnxn.SendThread#run

里面调用了 org.apache.zookeeper.ClientCnxn.EventThread#queueEvent 这个是往 event 队列放数据的。

org.apache.zookeeper.ClientCnxn.EventThread#run 这个就是从队列里面取数据了

```
public void run() {
   try {
      isRunning = true;
      while (true) {
         Object event = waitingEvents.take();
         if (event == eventOfDeath) {
            wasKilled = true;
         } else {
            processEvent(event);
         }
         if (wasKilled)
            synchronized (waitingEvents) {
               if (waitingEvents.isEmpty()) {
                  isRunning = false;
                  break;
               }
            }
      }
   } catch (InterruptedException e) {
      LOG.error("Event thread exiting due to interruption", e);
   }

    LOG.info("EventThread shut down for session: 0x{}",
             Long.toHexString(getSessionId()));
}
```