**Copycat把member分成3种**

**Active member：** 包含Raft协议中的follower, candidate, or leader 角色，即正常使用的member

**Passive member：**那些只是和集群其他的member保持同步，但不参加vote流程的member，称为Passive有passive节点的好处，是当需要加Active节点和替换fail的Active节点时，不需要catch up的过程，直接替换就可以passive节点，还可以作为只读节点

**Reserve member：**

对于比较大的集群，维护passive member的代价也很高；所以为了降低这个成本，又加入reserve member

//**FollowerState** 关键逻辑是等待超时，并试图成为candidator

public synchronized CompletableFuture<ServerState> open() {
return super.open().thenRun(this::startHeartbeatTimeout).thenApply(v -> this);
}

在open的时候，异步run，thenRun就是不依赖前面的输入 startHeartbeatTimeout –> resetHeartbeatTimeout

private void resetHeartbeatTimeout() {
// Set the election timeout in a semi-random fashion with the random range
// being election timeout and 2 \* election timeout.
Duration delay = context.getElectionTimeout().plus(Duration.ofMillis(random.nextInt((int) context.getElectionTimeout().toMillis()))); //随机产生delay时间
heartbeatTimer = context.getThreadContext().schedule(delay, () -> { //delay时间到后
heartbeatTimer = null;
if (isOpen()) {
context.setLeader(0); //清空leader
sendPollRequests(); //发送poll请求
}
});
}

当超时结束时，是否我就可以成为candidator，raft论文里面限制，必须要具有最新commit的member才能成为candidator，那么我怎么知道我是否具有最新的commit

发送poll请求，majority都同意，说明我的commit比大多数都要新或一样新，说明我具有最新的commit

注意这个resetHeartbeatTimeout，不光在这里调用，基本在Follower所有的请求响应时都会调用，即如果和leader有交互，会不停的重启这个timer

只有接收不到leader的心跳了，才会调用sendPollRequests，试图成为candidator

/\*\*
\* Polls all members of the cluster to determine whether this member should transition to the CANDIDATE state.
\*/
private void sendPollRequests() {
// Create a quorum that will track the number of nodes that have responded to the poll request.
final AtomicBoolean complete = new AtomicBoolean();
final Set<ServerMember> votingMembers = new HashSet<> (context.getClusterState().getActiveMemberStates().stream().map(MemberState::getMember).collect(Collectors.toList()));
//找到所有active的members，并且生成Quorum callback new Consumer<Boolean>() {}
final Quorum quorum = new Quorum(context.getClusterState().getQuorum(), (elected) -> {
// If a majority of the cluster indicated they would vote for us then transition to candidate.
complete.set(true);
if (elected) { //如果elected成功
context.transition(CopycatServer.State.CANDIDATE); //迁移到candidate
} else {
resetHeartbeatTimeout();
}
});
// Once we got the last log term, iterate through each current member
// of the cluster and vote each member for a vote.
for (ServerMember member : votingMembers) {
LOGGER.debug("{} - Polling {} for next term {}", context.getCluster().member().address(), member, context.getTerm() + 1);
PollRequest request = PollRequest.builder()
.withTerm(context.getTerm())
.withCandidate(context.getCluster().member().id())
.withLogIndex(lastIndex) //当前我的lastindex
.withLogTerm(lastTerm) //当前我的lastTerm，别人需要根据index和term来决定是否poll我
.build();
context.getConnections().getConnection(member.serverAddress()).thenAccept(connection -> {
connection.<PollRequest, PollResponse>send(request).whenCompleteAsync((response, error) -> { //异步发送request，并且加上callback
context.checkThread();
if (isOpen() && !complete.get()) {
if (error != null) {
LOGGER.warn("{} - {}", context.getCluster().member().address(), error.getMessage());
quorum.fail();
} else {
if (response.term() > context.getTerm()) {
context.setTerm(response.term());
}
if (!response.accepted()) {
LOGGER.debug("{} - Received rejected poll from {}", context.getCluster().member().address(), member);
quorum.fail();
} else if (response.term() != context.getTerm()) {
LOGGER.debug("{} - Received accepted poll for a different term from {}", context.getCluster().member().address(), member);
quorum.fail();
} else {
LOGGER.debug("{} - Received accepted poll from {}", context.getCluster().member().address(), member);
quorum.succeed(); //各种错误后，只有这个表示对方poll我了,触发callback
}
}
}
}, context.getThreadContext().executor());
});
}
}

**quorum法定人数**

public class Quorum {
private final int quorum;
private int succeeded = 1;
private int failed;
private Consumer<Boolean> callback;
private boolean complete;
public Quorum(int quorum, Consumer<Boolean> callback) {
this.quorum = quorum;
this.callback = callback;
}
private void checkComplete() {
if (!complete && callback != null) {
if (succeeded >= quorum) { //人数过半即可
complete = true;
callback.accept(true); //通过quorum.success()方法check回调callback
} else if (failed >= quorum) {
complete = true;
callback.accept(false);
}
}
}
/\*\*
\* Indicates that a call in the quorum succeeded.
\*/
public Quorum succeed() {
succeeded++;
checkComplete();
return this;
}

//**FollowerState**

@Override
public CompletableFuture<AppendResponse> append(AppendRequest request) {
CompletableFuture<AppendResponse> future = super.append(request);
// Reset the heartbeat timeout.
resetHeartbeatTimeout();
// Send AppendEntries requests to passive members if necessary.
appender.appendEntries();
return future;
}

可以看到除了调用super的append以及resetHB外，还有appender.appendEntries();这应该是Follower需要承担起，把数据同步给passive的责任

**CandidateState:**

candidate的逻辑就是通过vote，变成leader

public synchronized CompletableFuture<ServerState> open() {
return super.open().thenRun(this::startElection).thenApply(v -> this);
}

// startElection-> sendVoteRequests
private void sendVoteRequests() {
// When the election timer is reset, increment the current term and
// restart the election.
context.setTerm(context.getTerm() + 1).setLastVotedFor(context.getCluster().member().id());
//重新选举，所以term+1；setLastVoteFor，设成self，先投自己一票
final AtomicBoolean complete = new AtomicBoolean();
final Set<ServerMember> votingMembers = new HashSet<>(context.getClusterState().getActiveMemberStates().stream().map(MemberState::getMember).collect(Collectors.toList()));
// Send vote requests to all nodes. The vote request that is sent
// to this node will be automatically successful.
// First check if the quorum is null. If the quorum isn't null then that
// indicates that another vote is already going on.
final Quorum quorum = new Quorum(context.getClusterState().getQuorum(), (elected) -> {
complete.set(true);
if (elected) {
context.transition(CopycatServer.State.LEADER); //vote成功，就是leader
} else {
context.transition(CopycatServer.State.FOLLOWER);
}
});
// Once we got the last log term, iterate through each current member
// of the cluster and vote each member for a vote.
for (ServerMember member : votingMembers) {
LOGGER.debug("{} - Requesting vote from {} for term {}", context.getCluster().member().address(), member, context.getTerm());
VoteRequest request = VoteRequest.builder()
.withTerm(context.getTerm())
.withCandidate(context.getCluster().member().id())
.withLogIndex(lastIndex)
.withLogTerm(lastTerm)
.build();
context.getConnections().getConnection(member.serverAddress()).thenAccept(connection -> {
connection.<VoteRequest, VoteResponse>send(request).whenCompleteAsync((response, error) -> {
context.checkThread();
if (isOpen() && !complete.get()) {
if (error != null) {
LOGGER.warn(error.getMessage());
quorum.fail();
} else {
if (response.term() > context.getTerm()) {
LOGGER.debug("{} - Received greater term from {}", context.getCluster().member().address(), member);
context.setTerm(response.term());
complete.set(true);
context.transition(CopycatServer.State.FOLLOWER);
} else if (!response.voted()) {
LOGGER.debug("{} - Received rejected vote from {}", context.getCluster().member().address(), member);
quorum.fail();
} else if (response.term() != context.getTerm()) {
LOGGER.debug("{} - Received successful vote for a different term from {}", context.getCluster().member().address(), member);
quorum.fail();
} else {
LOGGER.debug("{} - Received successful vote from {}", context.getCluster().member().address(), member);
quorum.succeed();//callback触发
}
}
}
}, context.getThreadContext().executor());
});
}
}

响应append请求

public CompletableFuture<AppendResponse> append(AppendRequest request) {
context.checkThread();
// If the request indicates a term that is greater than the current term then
// assign that term and leader to the current context and step down as a candidate.
if (request.term() >= context.getTerm()) { //如果term比我的大，说明已经有leader
context.setTerm(request.term());
context.transition(CopycatServer.State.FOLLOWER); //退化成follower
}
return super.append(request);
}

响应vote请求

@Override
public CompletableFuture<VoteResponse> vote(VoteRequest request) {
context.checkThread();
logRequest(request);
// If the request indicates a term that is greater than the current term then
// assign that term and leader to the current context and step down as a candidate.
if (updateTermAndLeader(request.term(), 0)) { //如果request term比我大，说明已经有leader
CompletableFuture<VoteResponse> future = super.vote(request);
context.transition(CopycatServer.State.FOLLOWER); //退化成follower
return future;
}
// If the vote request is not for this candidate then reject the vote.
if (request.candidate() == context.getCluster().member().id()) { //否则，只有request的candidate id是我，我才同意
return CompletableFuture.completedFuture(logResponse(VoteResponse.builder()
.withStatus(Response.Status.OK)
.withTerm(context.getTerm())
.withVoted(true)
.build()));
} else { //candidate不会同意其他的candidate
return CompletableFuture.completedFuture(logResponse(VoteResponse.builder()
.withStatus(Response.Status.OK)
.withTerm(context.getTerm())
.withVoted(false)
.build()));
}
}

public class AtomixOne {

public static void main(String[] args){

//设置server\_1的地址和端口

Address address = new Address("127.0.0.1", 8001);

CopycatServer server = CopycatServer.*builder*(address)

.withStateMachine(ValueStateMachine::new)

.withTransport(NettyTransport.*builder*()

.withThreads(4)

.build())

.withStorage(Storage.*builder*()

.withDirectory(new File("I:\\elecate\_leader\\logs"))**//全局目录必须相同**

.withStorageLevel(StorageLevel.*DISK*)

.build())

.build();

server.serializer().register(PutCommand.class);

server.serializer().register(GetQuery.class);

//启动服务器

CompletableFuture<CopycatServer> future = server.bootstrap();

future.join();//挂起主线程

}

}

源代码跟进启动：激活先转变状态机Follower状态

**8001启动：**

17:10:02.238 [catalyst-event-loop-1] DEBUG i.n.handler.logging.LoggingHandler - [id: 0x75865e3a, L:/127.0.0.1:8001] ACTIVE

17:10:02.259 [copycat-server-/127.0.0.1:8001-copycat] INFO i.a.c.server.state.ServerContext - /127.0.0.1:8001 - Transitioning to FOLLOWER

17:10:03.585 [copycat-server-/127.0.0.1:8001-copycat] DEBUG i.a.c.server.state.FollowerState - /127.0.0.1:8001 - Heartbeat timed out in PT1.318S

17:10:03.588 [copycat-server-/127.0.0.1:8001-copycat] INFO i.a.c.server.state.ServerContext - /127.0.0.1:8001 - Transitioning to CANDIDATE

17:10:03.591 [copycat-server-/127.0.0.1:8001-copycat] INFO i.a.c.server.state.CandidateState - /127.0.0.1:8001 - Starting election

17:10:03.593 [copycat-server-/127.0.0.1:8001-copycat] DEBUG i.a.c.server.state.ServerContext - /127.0.0.1:8001 - Set term 1

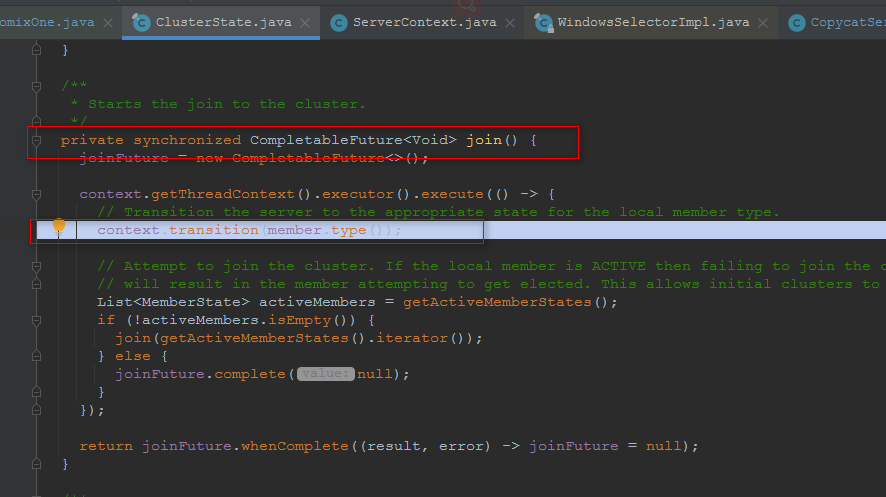
17:10:03.594 [copycat-server-/127.0.0.1:8001-copycat] DEBUG i.a.c.server.state.ServerContext - /127.0.0.1:8001 - Voted for /127.0.0.1:8001

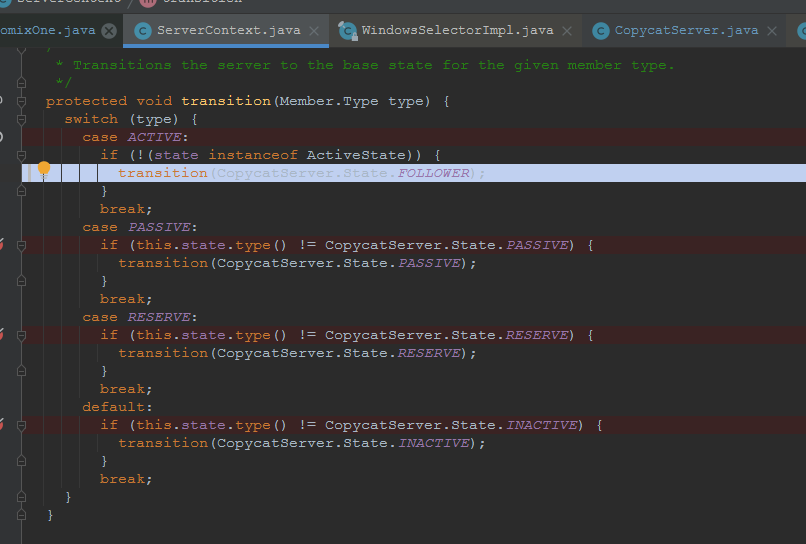
17:10:03.598 [copycat-server-/127.0.0.1:8001-copycat] INFO i.a.c.server.state.ServerContext - /127.0.0.1:8001 - Transitioning to LEADER

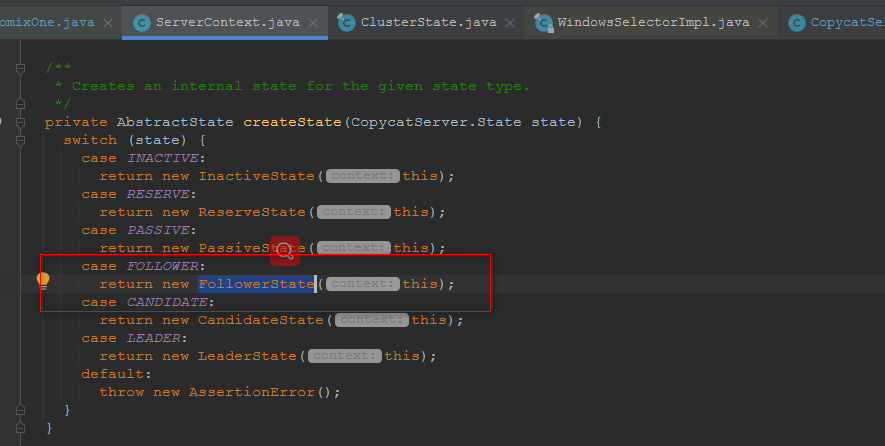
17:10:03.602 [copycat-server-/127.0.0.1:8001-copycat] DEBUG i.a.c.server.state.CandidateState - /127.0.0.1:8001 - Cancelling election

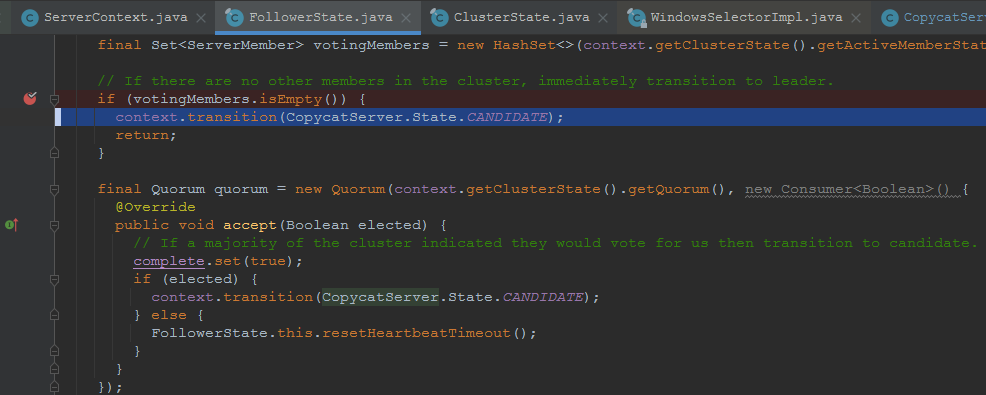
17:10:03.617 [copycat-server-/127.0.0.1:8001-copycat] INFO i.a.c.server.state.ServerContext - /127.0.0.1:8001 - Found leader /127.0.0.1:8001

17:10:03.637 [copycat-server-/127.0.0.1:8001-copycat] INFO i.a.copycat.server.CopycatServer - Server started successfully!

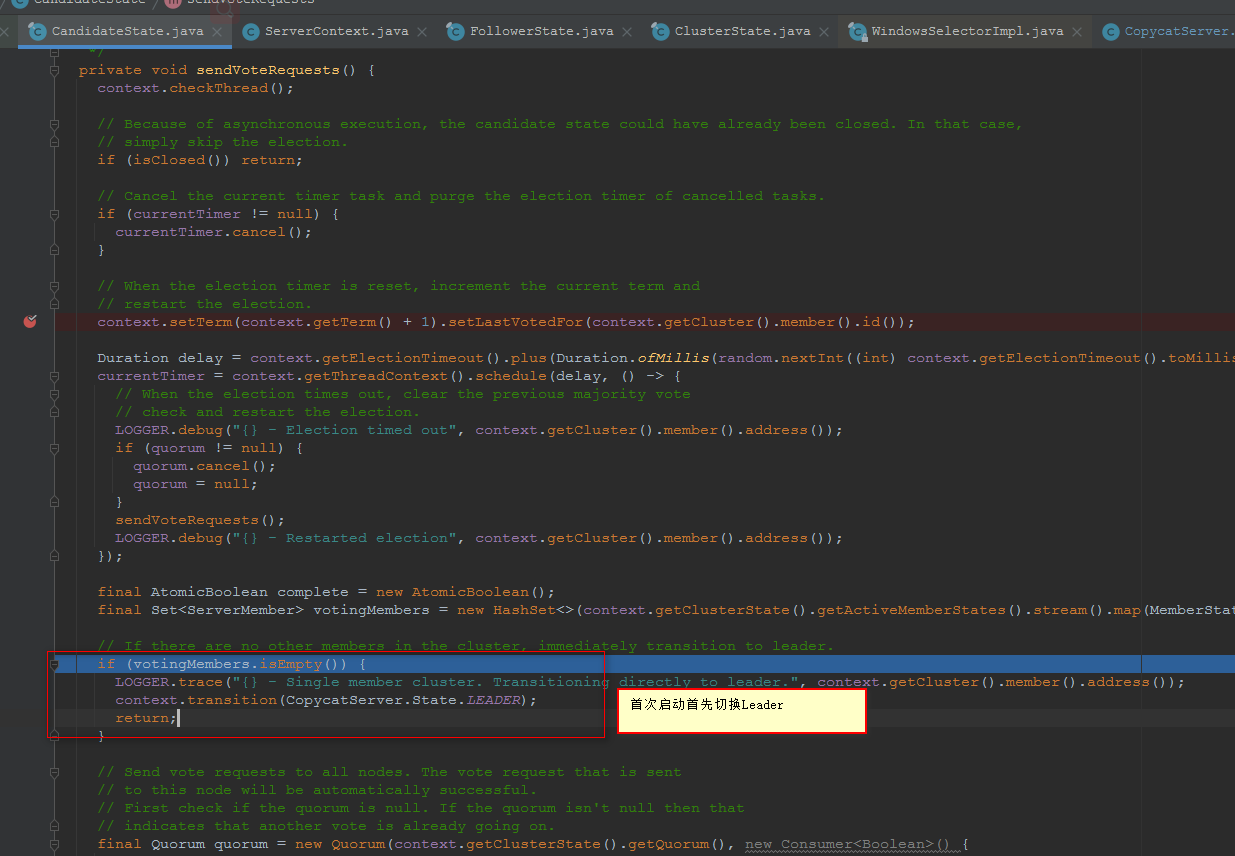








**CandidateState：**



**8002启动：**

19:47:44.832 [catalyst-event-loop-1] DEBUG i.n.handler.logging.LoggingHandler - [id: 0x18520c66, L:/127.0.0.1:8002] ACTIVE

19:47:44.836 [copycat-server-/127.0.0.1:8002-copycat] INFO i.a.c.server.state.ServerContext - /127.0.0.1:8002 - Transitioning to FOLLOWER

19:47:44.857 [copycat-server-/127.0.0.1:8002-copycat] DEBUG i.a.c.server.state.ClusterState - /127.0.0.1:8002 - Attempting to join via /127.0.0.1:8001

19:47:44.858 [copycat-server-/127.0.0.1:8002-copycat] INFO i.a.c.transport.netty.NettyClient - Connecting to /127.0.0.1:8001

19:47:44.888 [catalyst-event-loop-2] INFO i.a.c.transport.netty.NettyClient - Connected to /127.0.0.1:8001

19:47:44.909 [copycat-server-/127.0.0.1:8002-copycat] DEBUG io.netty.util.Recycler - -Dio.netty.recycler.maxCapacityPerThread: 32768

19:47:44.910 [copycat-server-/127.0.0.1:8002-copycat] DEBUG io.netty.util.Recycler - -Dio.netty.recycler.maxSharedCapacityFactor: 2

19:47:44.910 [copycat-server-/127.0.0.1:8002-copycat] DEBUG io.netty.util.Recycler - -Dio.netty.recycler.linkCapacity: 16

19:47:44.910 [copycat-server-/127.0.0.1:8002-copycat] DEBUG io.netty.util.Recycler - -Dio.netty.recycler.ratio: 8

19:47:44.923 [copycat-server-/127.0.0.1:8002-copycat] DEBUG io.netty.buffer.AbstractByteBuf - -Dio.netty.buffer.bytebuf.checkAccessible: true

19:47:44.926 [copycat-server-/127.0.0.1:8002-copycat] DEBUG i.n.util.ResourceLeakDetectorFactory - Loaded default ResourceLeakDetector: io.netty.util.ResourceLeakDetector@4c1473a

19:47:45.031 [catalyst-event-loop-1] DEBUG i.n.handler.logging.LoggingHandler - [id: 0x18520c66, L:/127.0.0.1:8002] RECEIVED: [id: 0x602e8f9c, L:/127.0.0.1:8002 - R:/127.0.0.1:50658]

19:47:45.062 [copycat-server-/127.0.0.1:8002-copycat] INFO i.a.c.server.state.ServerContext - /127.0.0.1:8002 - Found leader /127.0.0.1:8001

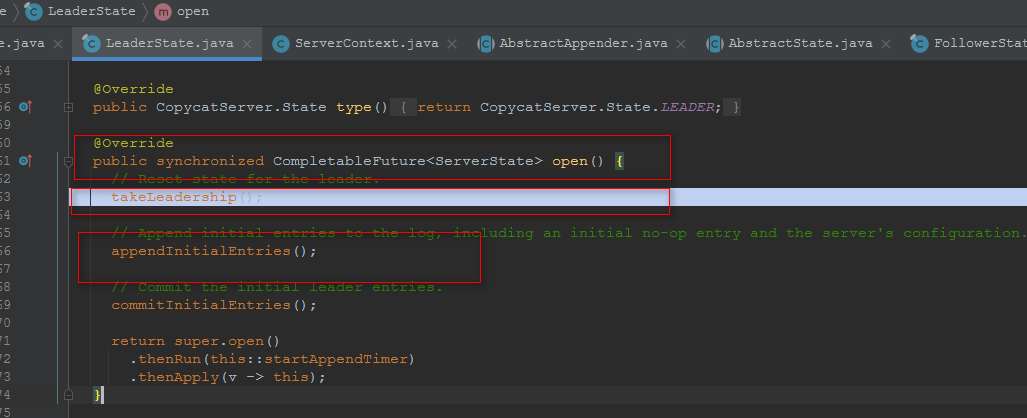
19:47:45.063 [copycat-server-/127.0.0.1:8002-copycat] DEBUG i.a.c.server.state.ClusterState - /127.0.0.1:8002 - Sending server identification to /127.0.0.1:8001

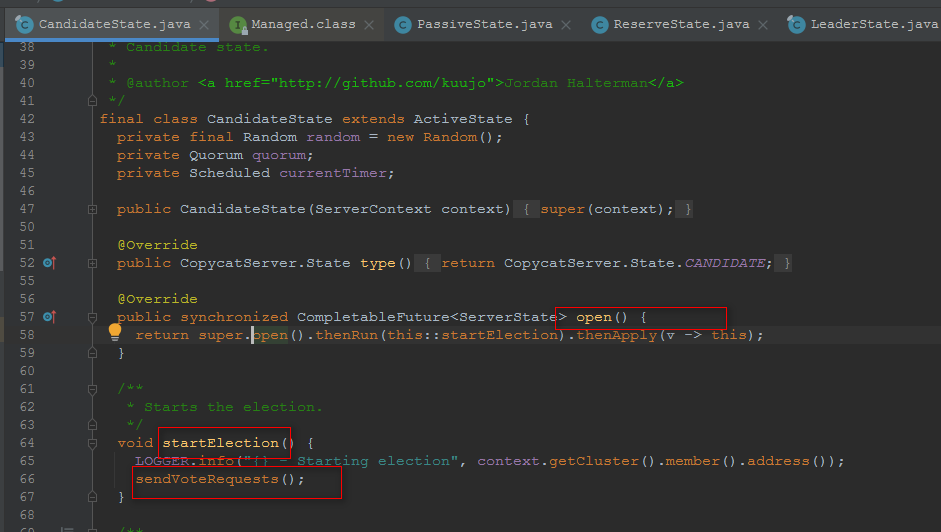
19:47:45.099 [copycat-server-/127.0.0.1:8002-copycat] DEBUG i.a.c.server.state.ClusterState - /127.0.0.1:8002 - Failed to update configuration: configuration change already in progress

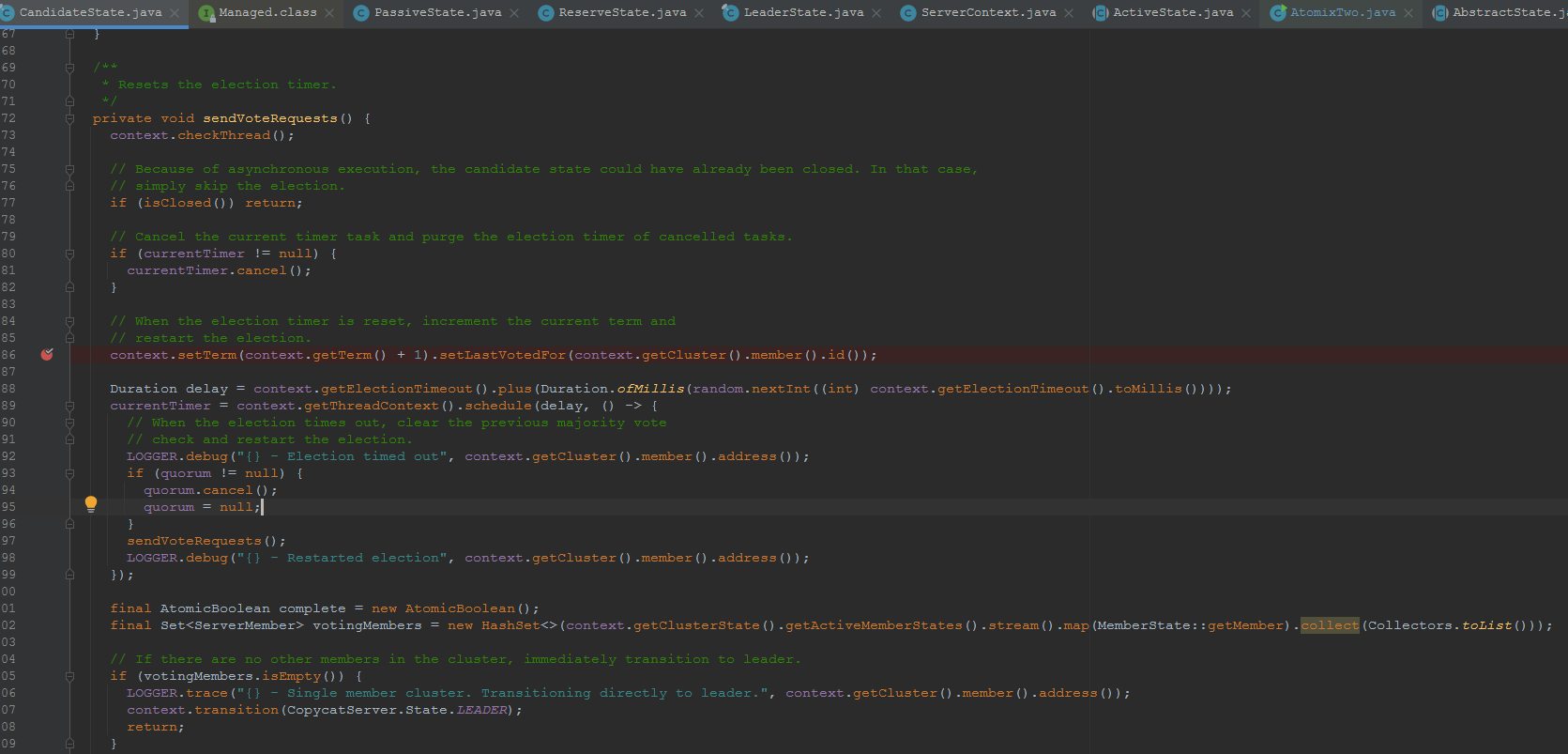
19:47:45.116 [copycat-server-/127.0.0.1:8002-copycat] DEBUG i.a.c.server.state.FollowerState - /127.0.0.1:8002 - Rejected AppendRequest[term=1, leader=2130715285, logIndex=3, logTerm=1, entries=[0], commitIndex=2, globalIndex=2]: Previous index (3) is greater than the local log's last index (2)

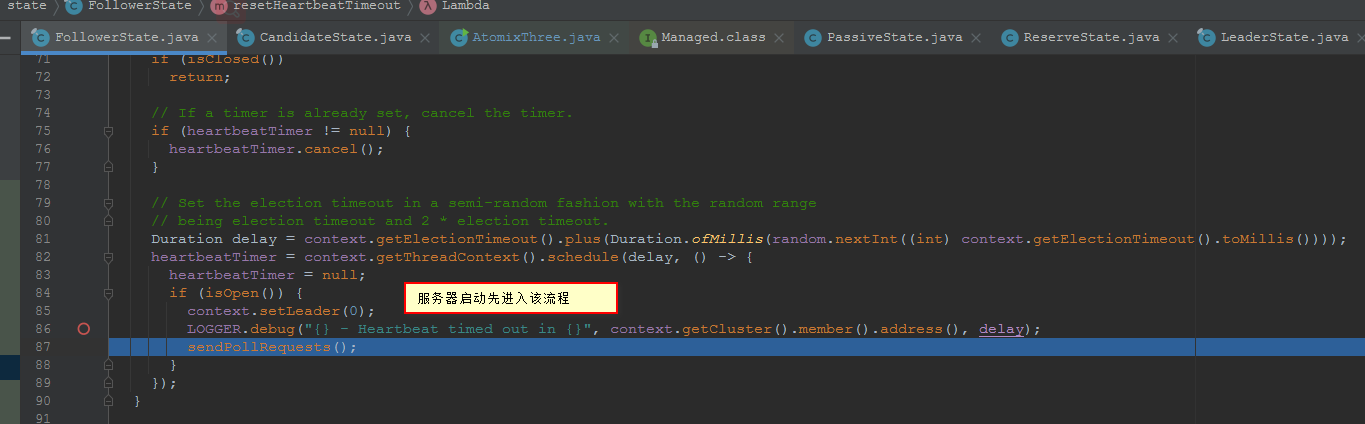
19:47:45.134 [copycat-server-/127.0.0.1:8002-copycat] INFO i.a.c.server.state.ClusterState - /127.0.0.1:8002 - Successfully joined via /127.0.0.1:8001

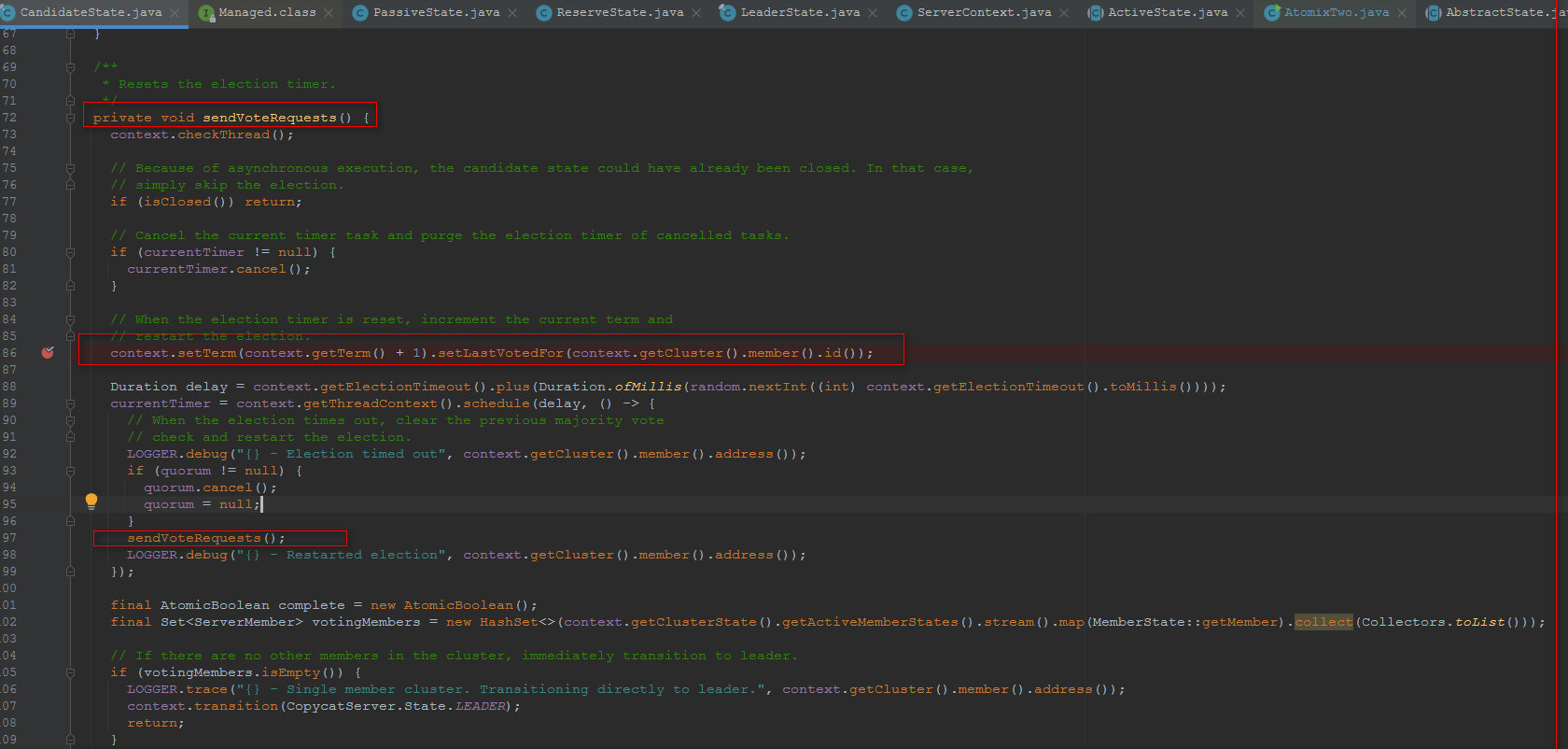
19:47:45.134 [copycat-server-/127.0.0.1:8002-copycat] INFO i.a.copycat.server.CopycatServer - Server started successfully!



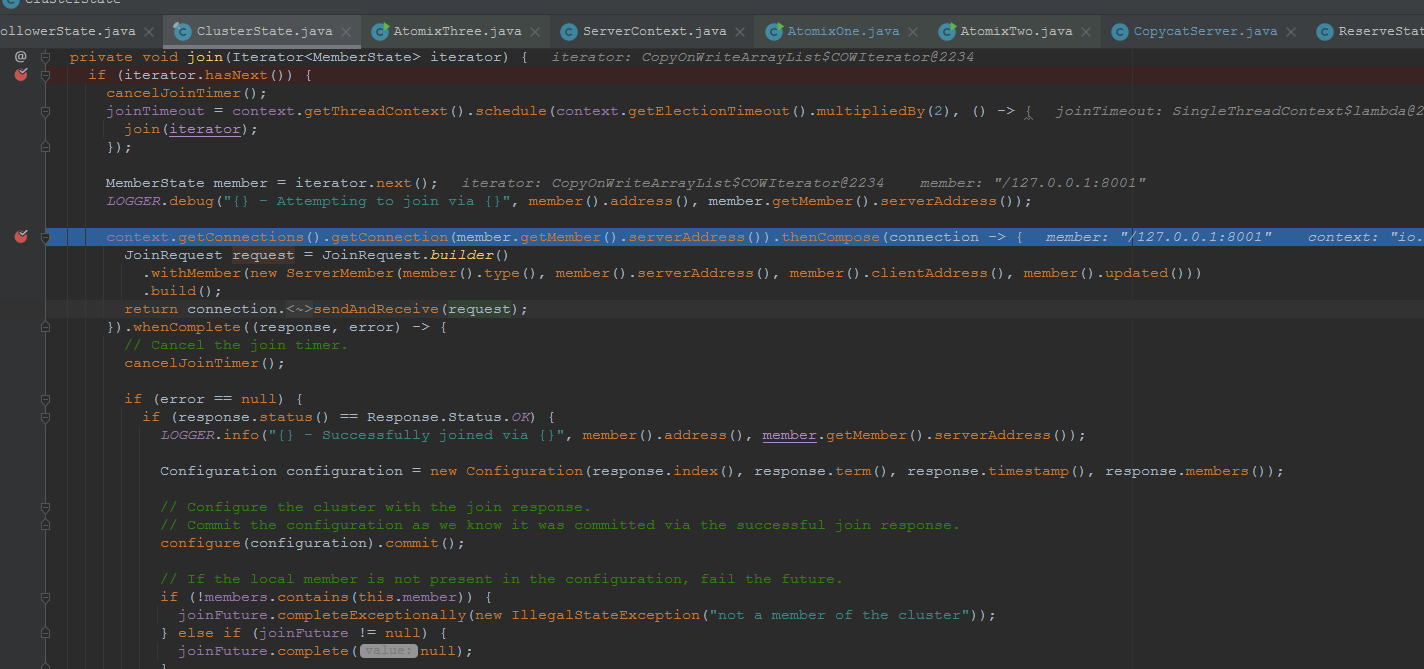


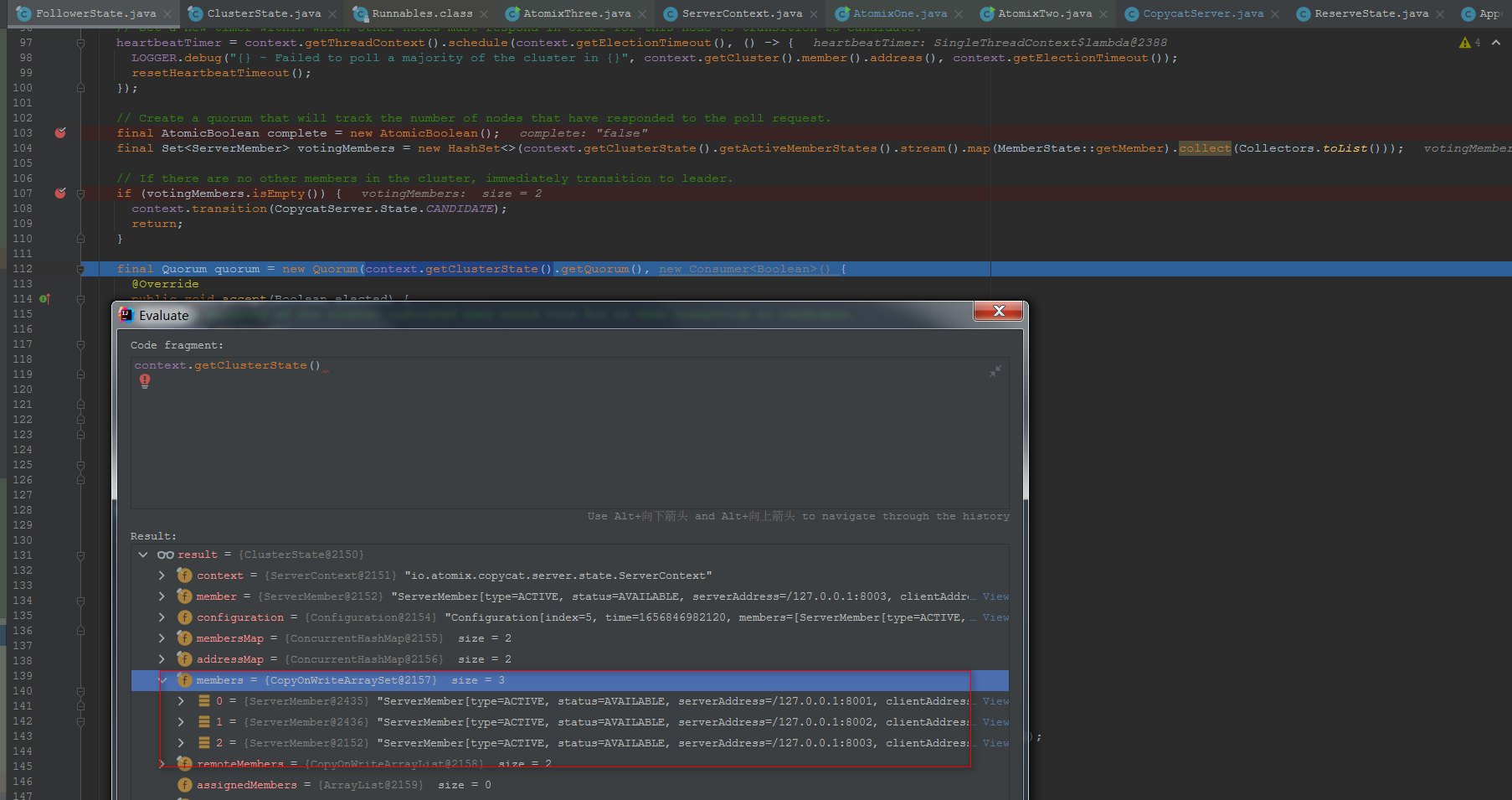






**tcp连接leader ip**





**请求其它客户端给自己投票**

