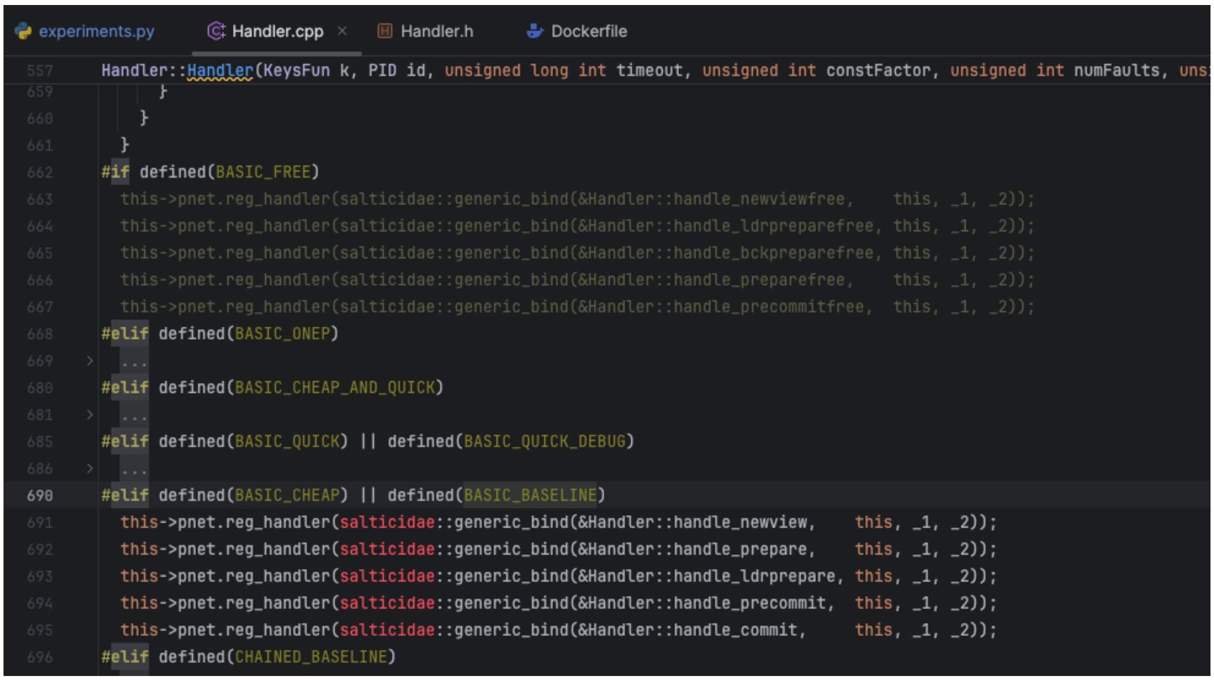
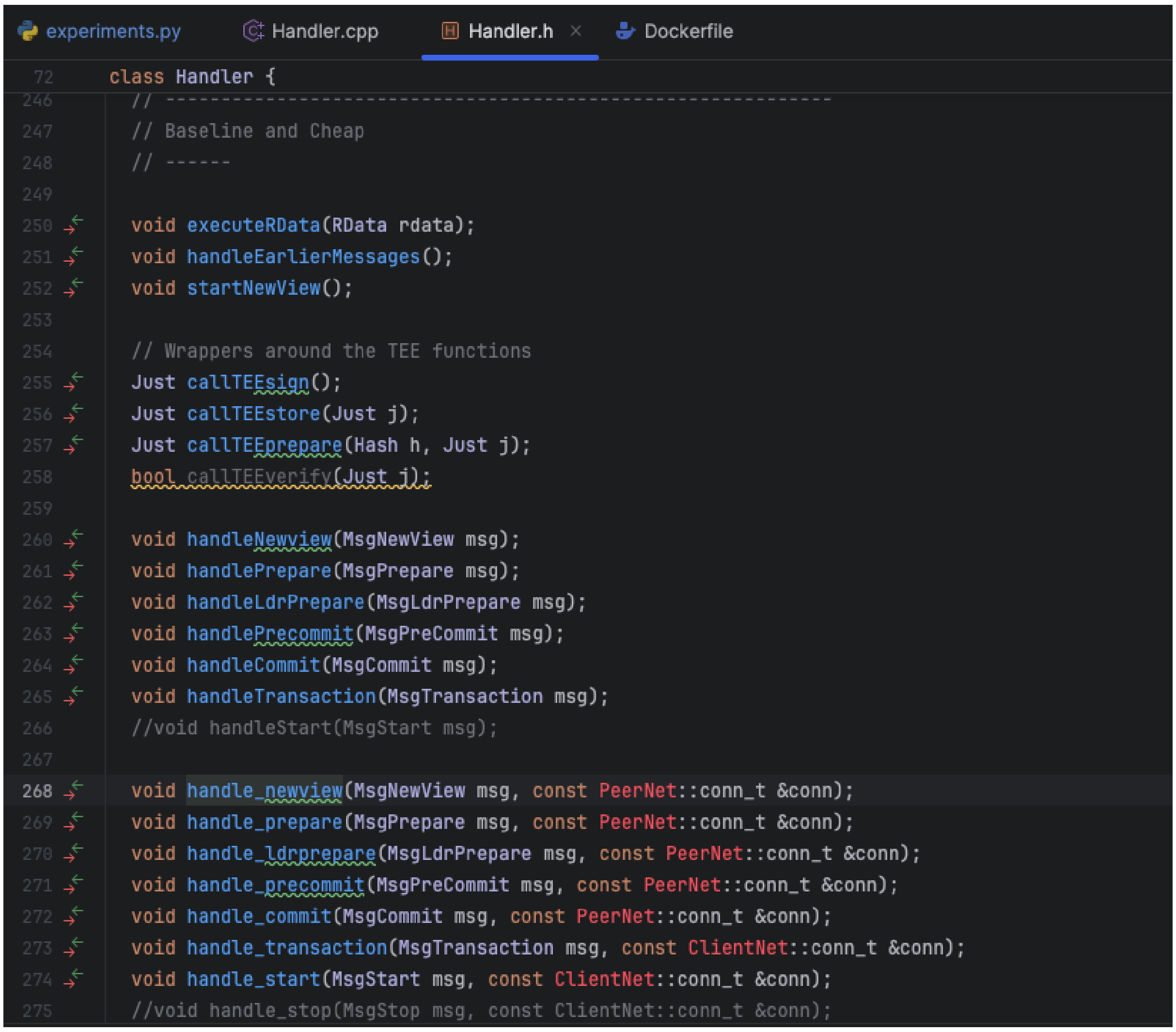
Hotstuff → Code Analysis

→ Declaration of Hotstuff protocol steps



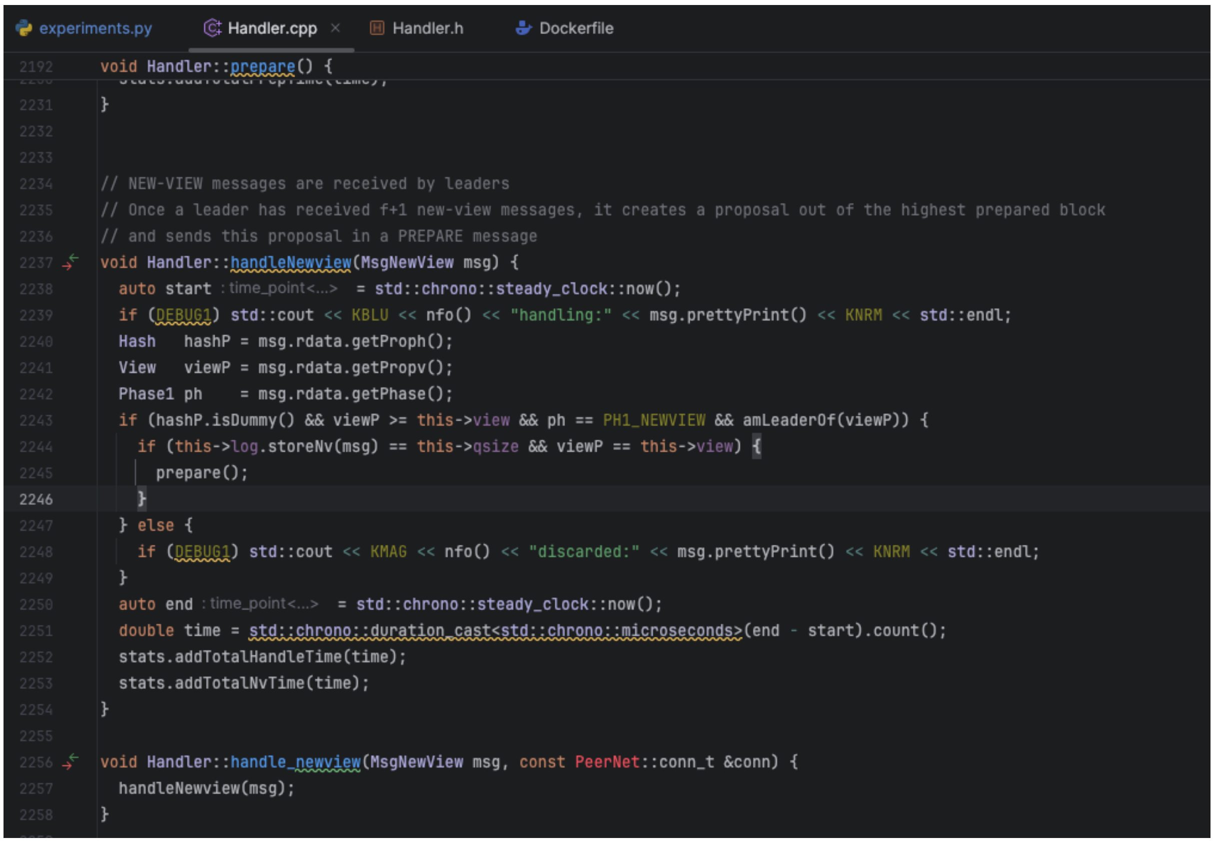
```cpp
Handler::Handler(KeysFun k, PID id, unsigned long int timeout, unsigned int constFactor, unsigned int numFaults, uns
        }
      }
    }
#if defined(BASIC_FREE)
  this->pnet.reg_handler(salticidae::generic_bind(&Handler::handle_newviewfree,    this, _1, _2));
  this->pnet.reg_handler(salticidae::generic_bind(&Handler::handle_ldrpreparefree, this, _1, _2));
  this->pnet.reg_handler(salticidae::generic_bind(&Handler::handle_bckpreparefree, this, _1, _2));
  this->pnet.reg_handler(salticidae::generic_bind(&Handler::handle_preparefree,    this, _1, _2));
  this->pnet.reg_handler(salticidae::generic_bind(&Handler::handle_precommitfree,  this, _1, _2));
#elif defined(BASIC_ONEP)
  ...
#elif defined(BASIC_CHEAP_AND_QUICK)
  ...
#elif defined(BASIC_QUICK) || defined(BASIC_QUICK_DEBUG)
  ...
#elif defined(BASIC_CHEAP) || defined(BASIC_BASELINE)
  this->pnet.reg_handler(salticidae::generic_bind(&Handler::handle_newview,    this, _1, _2));
  this->pnet.reg_handler(salticidae::generic_bind(&Handler::handle_prepare,    this, _1, _2));
  this->pnet.reg_handler(salticidae::generic_bind(&Handler::handle_ldrprepare, this, _1, _2));
  this->pnet.reg_handler(salticidae::generic_bind(&Handler::handle_precommit,  this, _1, _2));
  this->pnet.reg_handler(salticidae::generic_bind(&Handler::handle_commit,     this, _1, _2));
#elif defined(CHAINED_BASELINE)
```

→ Methods used for Basic Hotstuff



```cpp
class Handler {
    // --------------------------------------------------------------
    // Baseline and Cheap
    // ------

    void executeRData(RData rdata);
    void handleEarlierMessages();
    void startNewView();

    // Wrappers around the TEE functions
    Just callTEEsign();
    Just callTEEstore(Just j);
    Just callTEEprepare(Hash h, Just j);
    bool callTEEverify(Just j);

    void handleNewview(MsgNewView msg);
    void handlePrepare(MsgPrepare msg);
    void handleLdrPrepare(MsgLdrPrepare msg);
    void handlePrecommit(MsgPreCommit msg);
    void handleCommit(MsgCommit msg);
    void handleTransaction(MsgTransaction msg);
    //void handleStart(MsgStart msg);

    void handle_newview(MsgNewView msg, const PeerNet::conn_t &conn);
    void handle_prepare(MsgPrepare msg, const PeerNet::conn_t &conn);
    void handle_ldrprepare(MsgLdrPrepare msg, const PeerNet::conn_t &conn);
    void handle_precommit(MsgPreCommit msg, const PeerNet::conn_t &conn);
    void handle_commit(MsgCommit msg, const PeerNet::conn_t &conn);
    void handle_transaction(MsgTransaction msg, const ClientNet::conn_t &conn);
    void handle_start(MsgStart msg, const ClientNet::conn_t &conn);
    //void handle_stop(MsgStop msg, const ClientNet::conn_t &conn);
```

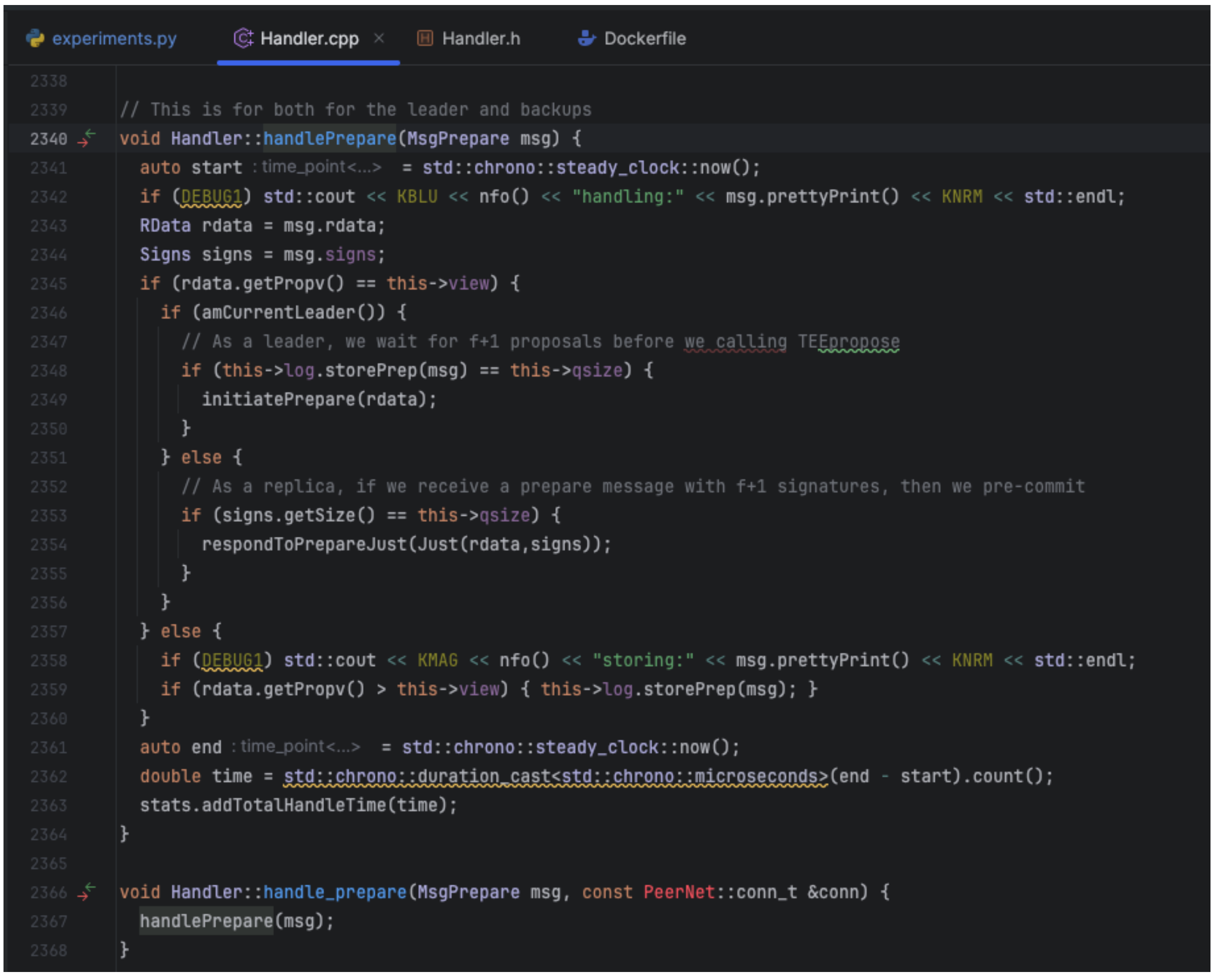
Phase 1: newview



```
2192  void Handler::prepare() {
2231  }
2232
2233
2234  // NEW-VIEW messages are received by leaders
2235  // Once a leader has received f+1 new-view messages, it creates a proposal out of the highest prepared block
2236  // and sends this proposal in a PREPARE message
2237  void Handler::handleNewview(MsgNewView msg) {
2238    auto start : time_point<...>  = std::chrono::steady_clock::now();
2239    if (DEBUG1) std::cout << KBLU << nfo() << "handling:" << msg.prettyPrint() << KNRM << std::endl;
2240    Hash   hashP = msg.rdata.getProph();
2241    View   viewP = msg.rdata.getPropv();
2242    Phase1 ph    = msg.rdata.getPhase();
2243    if (hashP.isDummy() && viewP >= this->view && ph == PH1_NEWVIEW && amLeaderOf(viewP)) {
2244      if (this->log.storeNv(msg) == this->qsize && viewP == this->view) {
2245        prepare();
2246      }
2247    } else {
2248      if (DEBUG1) std::cout << KMAG << nfo() << "discarded:" << msg.prettyPrint() << KNRM << std::endl;
2249    }
2250    auto end : time_point<...>  = std::chrono::steady_clock::now();
2251    double time = std::chrono::duration_cast<std::chrono::microseconds>(end - start).count();
2252    stats.addTotalHandleTime(time);
2253    stats.addTotalNvTime(time);
2254  }
2255
2256  void Handler::handle_newview(MsgNewView msg, const PeerNet::conn_t &conn) {
2257    handleNewview(msg);
2258  }
```

Check for f + 1 new-view messages → line 2244

Phase 2: prepare

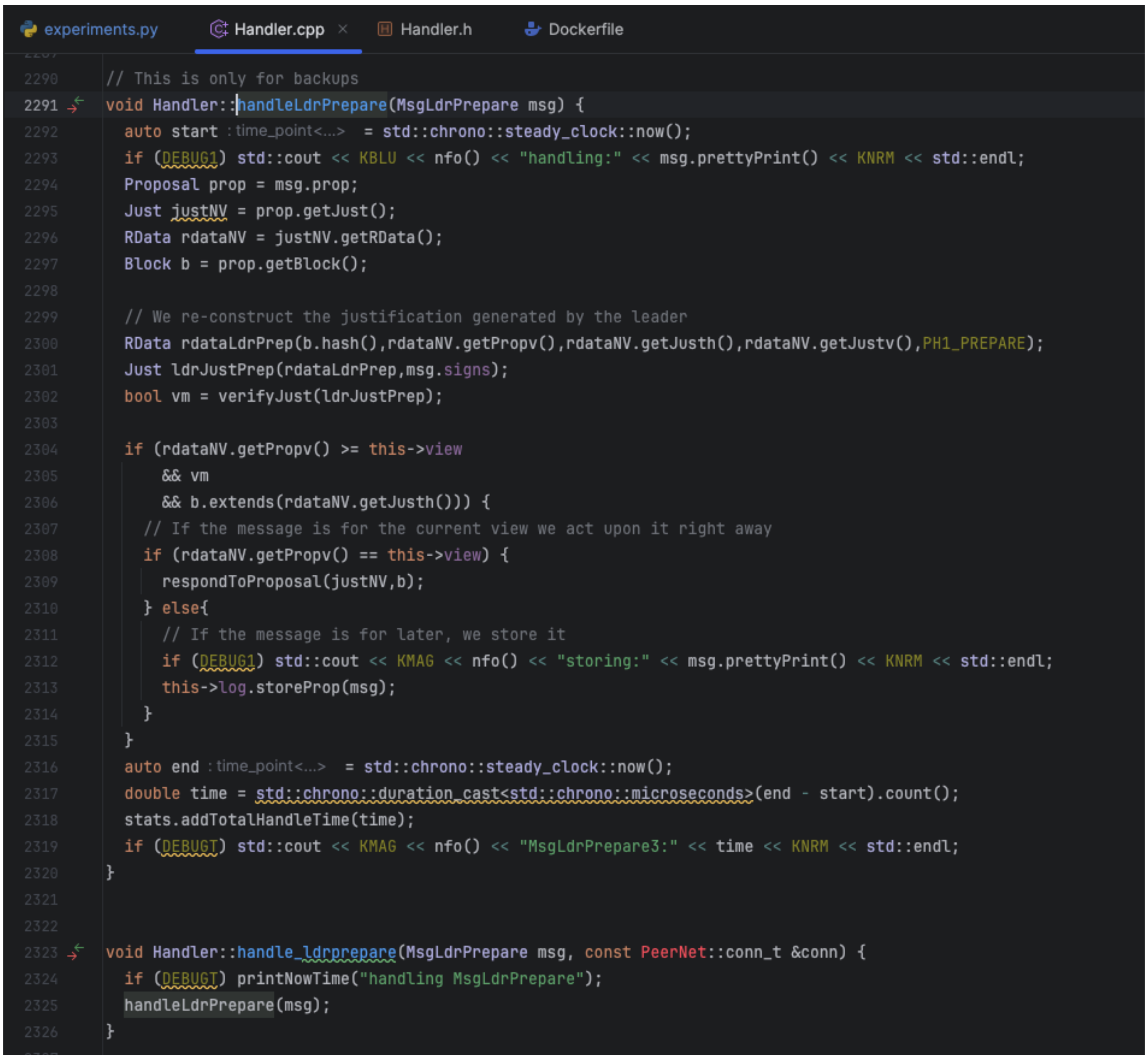


```cpp
2338
2339  // This is for both for the leader and backups
2340  void Handler::handlePrepare(MsgPrepare msg) {
2341    auto start : time_point<...>  = std::chrono::steady_clock::now();
2342    if (DEBUG1) std::cout << KBLU << nfo() << "handling:" << msg.prettyPrint() << KNRM << std::endl;
2343    RData rdata = msg.rdata;
2344    Signs signs = msg.signs;
2345    if (rdata.getPropv() == this->view) {
2346      if (amCurrentLeader()) {
2347        // As a leader, we wait for f+1 proposals before we calling TEEpropose
2348        if (this->log.storePrep(msg) == this->qsize) {
2349          initiatePrepare(rdata);
2350        }
2351      } else {
2352        // As a replica, if we receive a prepare message with f+1 signatures, then we pre-commit
2353        if (signs.getSize() == this->qsize) {
2354          respondToPrepareJust(Just(rdata,signs));
2355        }
2356      }
2357    } else {
2358      if (DEBUG1) std::cout << KMAG << nfo() << "storing:" << msg.prettyPrint() << KNRM << std::endl;
2359      if (rdata.getPropv() > this->view) { this->log.storePrep(msg); }
2360    }
2361    auto end : time_point<...>  = std::chrono::steady_clock::now();
2362    double time = std::chrono::duration_cast<std::chrono::microseconds>(end - start).count();
2363    stats.addTotalHandleTime(time);
2364  }
2365
2366  void Handler::handle_prepare(MsgPrepare msg, const PeerNet::conn_t &conn) {
2367    handlePrepare(msg);
2368  }
```

Check for f + 1 proposals/signatures

- **leader → line 2348**
- **replica → line 2353**

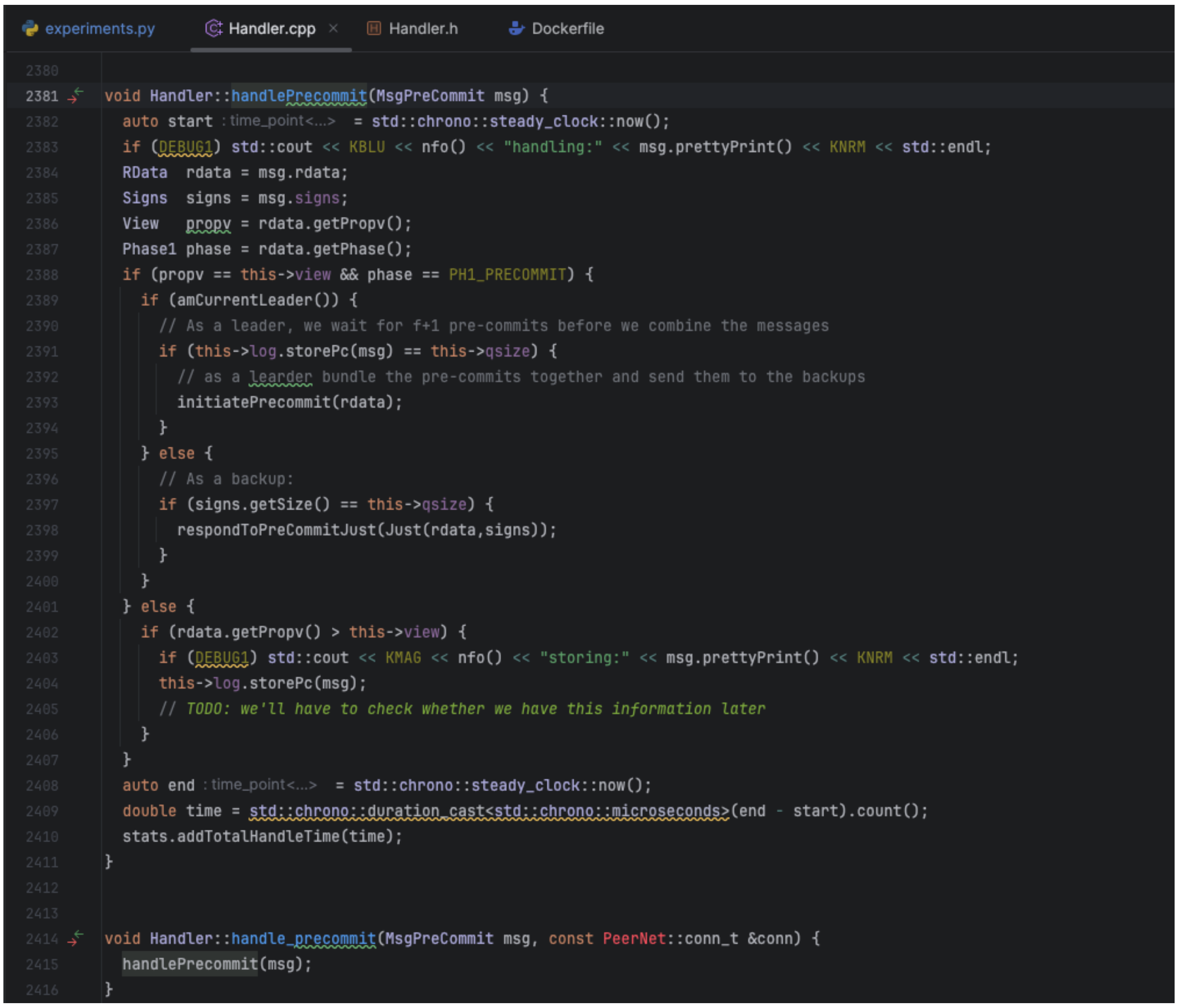
Phase 3: ldr prepare



```cpp
// This is only for backups
void Handler::handleLdrPrepare(MsgLdrPrepare msg) {
  auto start : time_point<...>  = std::chrono::steady_clock::now();
  if (DEBUG1) std::cout << KBLU << nfo() << "handling:" << msg.prettyPrint() << KNRM << std::endl;
  Proposal prop = msg.prop;
  Just justNV = prop.getJust();
  RData rdataNV = justNV.getRData();
  Block b = prop.getBlock();

  // We re-construct the justification generated by the leader
  RData rdataLdrPrep(b.hash(),rdataNV.getPropv(),rdataNV.getJusth(),rdataNV.getJustv(),PH1_PREPARE);
  Just ldrJustPrep(rdataLdrPrep,msg.signs);
  bool vm = verifyJust(ldrJustPrep);

  if (rdataNV.getPropv() >= this->view
      && vm
      && b.extends(rdataNV.getJusth())) {
    // If the message is for the current view we act upon it right away
    if (rdataNV.getPropv() == this->view) {
      respondToProposal(justNV,b);
    } else{
      // If the message is for later, we store it
      if (DEBUG1) std::cout << KMAG << nfo() << "storing:" << msg.prettyPrint() << KNRM << std::endl;
      this->log.storeProp(msg);
    }
  }
  auto end : time_point<...>  = std::chrono::steady_clock::now();
  double time = std::chrono::duration_cast<std::chrono::microseconds>(end - start).count();
  stats.addTotalHandleTime(time);
  if (DEBUGT) std::cout << KMAG << nfo() << "MsgLdrPrepare3:" << time << KNRM << std::endl;
}


void Handler::handle_ldrprepare(MsgLdrPrepare msg, const PeerNet::conn_t &conn) {
  if (DEBUGT) printNowTime("handling MsgLdrPrepare");
  handleLdrPrepare(msg);
}
```

No step that checks for a quorum formation.

Phase 4: pre-commit



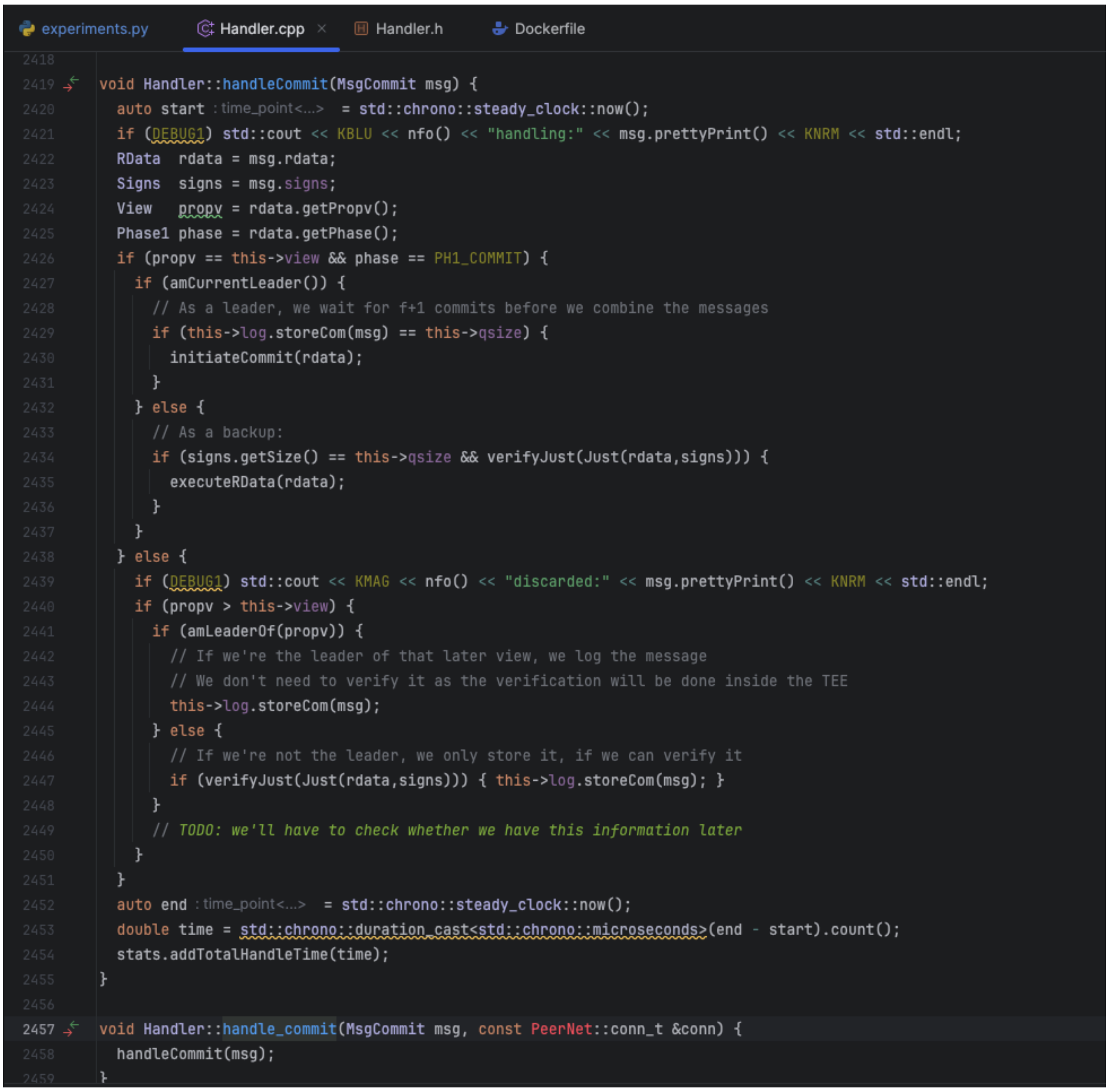
```
2380
2381 void Handler::handlePrecommit(MsgPreCommit msg) {
2382   auto start :time_point<...> = std::chrono::steady_clock::now();
2383   if (DEBUG1) std::cout << KBLU << nfo() << "handling:" << msg.prettyPrint() << KNRM << std::endl;
2384   RData  rdata = msg.rdata;
2385   Signs  signs = msg.signs;
2386   View   propv = rdata.getPropv();
2387   Phase1 phase = rdata.getPhase();
2388   if (propv == this->view && phase == PH1_PRECOMMIT) {
2389     if (amCurrentLeader()) {
2390       // As a leader, we wait for f+1 pre-commits before we combine the messages
2391       if (this->log.storePc(msg) == this->qsize) {
2392         // as a learder bundle the pre-commits together and send them to the backups
2393         initiatePrecommit(rdata);
2394       }
2395     } else {
2396       // As a backup:
2397       if (signs.getSize() == this->qsize) {
2398         respondToPreCommitJust(Just(rdata,signs));
2399       }
2400     }
2401   } else {
2402     if (rdata.getPropv() > this->view) {
2403       if (DEBUG1) std::cout << KMAG << nfo() << "storing:" << msg.prettyPrint() << KNRM << std::endl;
2404       this->log.storePc(msg);
2405       // TODO: we'll have to check whether we have this information later
2406     }
2407   }
2408   auto end :time_point<...> = std::chrono::steady_clock::now();
2409   double time = std::chrono::duration_cast<std::chrono::microseconds>(end - start).count();
2410   stats.addTotalHandleTime(time);
2411 }
2412
2413
2414 void Handler::handle_precommit(MsgPreCommit msg, const PeerNet::conn_t &conn) {
2415   handlePrecommit(msg);
2416 }
```

Check for f + 1 proposals/signatures

leader → line 2391

replica → line 2397

Phase 5: commit



```cpp
2418
2419  void Handler::handleCommit(MsgCommit msg) {
2420    auto start :time_point<...>  = std::chrono::steady_clock::now();
2421    if (DEBUG1) std::cout << KBLU << nfo() << "handling:" << msg.prettyPrint() << KNRM << std::endl;
2422    RData  rdata = msg.rdata;
2423    Signs  signs = msg.signs;
2424    View   propv = rdata.getPropv();
2425    Phase1 phase = rdata.getPhase();
2426    if (propv == this->view && phase == PH1_COMMIT) {
2427      if (amCurrentLeader()) {
2428        // As a leader, we wait for f+1 commits before we combine the messages
2429        if (this->log.storeCom(msg) == this->qsize) {
2430          initiateCommit(rdata);
2431        }
2432      } else {
2433        // As a backup:
2434        if (signs.getSize() == this->qsize && verifyJust(Just(rdata,signs))) {
2435          executeRData(rdata);
2436        }
2437      }
2438    } else {
2439      if (DEBUG1) std::cout << KMAG << nfo() << "discarded:" << msg.prettyPrint() << KNRM << std::endl;
2440      if (propv > this->view) {
2441        if (amLeaderOf(propv)) {
2442          // If we're the leader of that later view, we log the message
2443          // We don't need to verify it as the verification will be done inside the TEE
2444          this->log.storeCom(msg);
2445        } else {
2446          // If we're not the leader, we only store it, if we can verify it
2447          if (verifyJust(Just(rdata,signs))) { this->log.storeCom(msg); }
2448        }
2449        // TODO: we'll have to check whether we have this information later
2450      }
2451    }
2452    auto end :time_point<...>  = std::chrono::steady_clock::now();
2453    double time = std::chrono::duration_cast<std::chrono::microseconds>(end - start).count();
2454    stats.addTotalHandleTime(time);
2455  }
2456
2457  void Handler::handle_commit(MsgCommit msg, const PeerNet::conn_t &conn) {
2458    handleCommit(msg);
2459  }
```

Check for f + 1 proposals/signatures

- **leader → line 2428**
- **replica → line 2434**