```cpp
1  void run_async_flow_manager(){
2    repeat([this]{
3      return _manager.on_new_async_flow().then([this](){
4        auto af= _manager.cur_async_flow();
5        do_with(malware_detector(std::move(af)),
6          [](malware_detector&nf){
7            nf.events_registration();
8            return nf.run();
9          });
10       return stop_iteration::no;
11     ...
```

```cpp
14 future<>malware_detector::run(){
15   return _af.register_loop_fn([this](){
16     ...
17     if(_af.preprocessor.on_file_hash_event()){
18       // Get file hash from preprocessor.
19       auto _hash=_af.preprocessor.get_file_hash();
20       return _db.lookup(_hash).then([this](db_res res){
21         if(malware_detected_in_db(res)){
22           // File hash is stored in local database.
23           record_detected_malware();
24           return ready_future(action::drop);
25         }
26         // File hash is not stored in local database.
27         string name=build_up_name(_hash);
28         return _dns.query(name).then([this](dns_res res){
29           if(malware_detected_in_mhr(res)){
30             // File hash is presented in MHR.
31             record_detected_malware();
32             return ready_future(action::drop);
33           }
34           // File passes malware detection.
35           return ready_future(action::foreward);
36         });
37       }).then_wrapped([this](future<action>f){
38         try {
39           return f.get();
40         }
41         catch(...) {
42           // Either database or DNS query timeouts.
43           record_malware_detection_error();
44           return ready_future(action::drop);
45         }
46     ...
```

Flow Preprocessor

Local database

Continuation

Future<action::drop>

MHR

Continuation

Future<action::drop>

Future<action::foreward>

Future<action::drop>