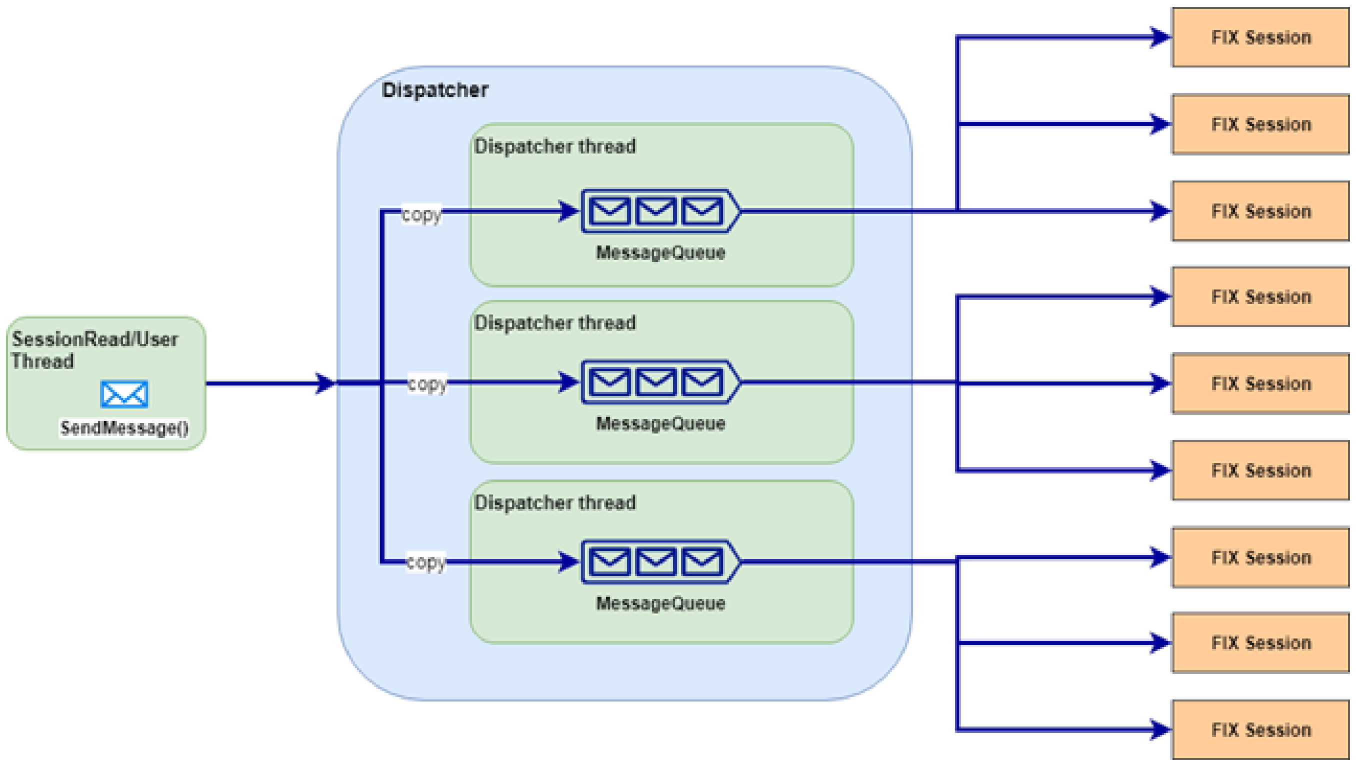
How to implement fan-out case

Fan-out case (delivery message to several recipients) is a standard task for market data routing. Such mechanism may be used for other solutions too.

FIX Antenna Java allows implementing effective fan-out solutions.

The main idea of effective implementation with FIX Antenna Java is reducing simultaneous access to outgoing sessions. Such an approach leads to getting the best performance. All outgoing sessions should be divided into few groups. Each such group is served by a separate worker thread. The original message should be copied to each target worker.
Such logic can be encapsulated into separate Dispatcher component:



You can find fan-out sample in FIX Antenna Java package (benchmarks\src\main\java\com\epam\benchmark\fanout\server\).

Dispatcher has access to queues of worker threads and can copy the original message to the necessary queues, which are related to required destinations sessions:

```
public void fanoutToAll(final FIXFieldList message) {
        int copies = getThreadCount() < outSessions.size() ? getThreadCount() : outSessions.size();
        for (int i = 0; i < copies; i++) {
                FIXFieldList safeMsgCopy = FIXFieldListFactory.newInstanceFromPool();
                message.deepCopyTo(safeMsgCopy);
                queues[i].push(safeMsgCopy);
        }
}
```

To reduce memory allocations, Dispatcher copy the original message to the message instance from the object pool. Such tactic provides faster processing than regular creating of new instance and helps to build 0GC applications.

In case of simplest delivering a message to all destination sessions, each working thread get own message copy from the queue and resend it to the sessions, which it is responsible for

```
FIXFieldList message = queues[worker].pop();
//fan-out message to worker's session range
for (int j = start; j < end; j++) {
        try {
                FIXSession session = outSessions.get(j);
                if (session != null) {
                        session.sendMessage(message);
                }
        } catch (Exception e) {
        LOG.warn(e.getMessage(), e);
}
```

Example of fan-out benchmark in FIXAJ

FIX Antenna package provides an example of such fan-out implementation in benchmark subdirectory. This implementation demonstrates the works of a fan-out gateway and provides mechanisms for measuring its performance.

The source code of fan-out benchmark is available in *benchmarks/src/com/epam/benchmark/fanout* subfolder of the package. It consists from such elements:

- *com.epam.benchmark.fanout.FanoutConfiguration* - constants, which are used in benchmark implementation;
- *com.epam.benchmark.fanout.client.MDProducerClient* - producers of test FIX messages. It simulates Exchanges, which send *'W'* messages to the fan-out router;
- *com.epam.benchmark.fanout.client.MDConsumerClient* - consumers of messages from the fan-out server. Simulates Traders, which consume *'W'* messages for possible decision making;
- *com.epam.benchmark.fanout.server.FanoutServer* - fan-out router implementation, which receives messages from produces and route them to consumers;
- *com.epam.benchmark.fanout.server.MessageDispatcher* and *MessageQueue* - Dispatcher component implementation;
- *com.epam.benchmark.fanout.BenchmarkReporter* - program, which can parse pcap files and provide a performance report;

The scripts to run fan-out programs and measure performance are available in the *benchmarks* subfolder of the package:

- *runFanoutServerWithTCPDump.sh* - script to run fan-out server with pcap file recording;
- *runFanoutConsumers.sh* - script to run fan-out consumers;
- *runFanoutProducers.sh* - script to run fan-out producers;
- *runFanoutReporter.sh* - script to run performance report generation from pcap file;

The logger and FIXAJ configuration files for all components there are in *benchmarks/etc/benchmark/fanout* subfolder of the package.

The test message, which is used for routing, there is also inside the package: *benchmarks/etc/benchmark/fanout/producer/snapshot*.fix

How to run fan-out benchmark in FIXAJ

1. Unpack FIX Antenna Java package (fixaj-engine-full-X.Y.Z.zip);

2. Run fan-out server (Gateway simulator): *`./runFanoutServerWithTCPDump.sh`*

   ```
   $> ./runFanoutServerWithTCPDump.sh
   tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 5000 bytes
   Launching 5 message dispatcher workers
   [INFO ][SLF4JLogFactory]: SLF4J logger wrapper initialized
   [INFO ][ClassPathResourceLoader]: Load resource:/home/bmtest/fixaj/benchmarks/etc/benchmark/fanout/server
   /fixengine.properties
   ...
   [INFO ][FIXServer]: Server started on port 3000
   [INFO ][FIXServer]: Server started on port 3001
   ```

   NOTE: please make sure that user has permissions to collect TCP dumps.

3. Run consumer clients (Traders simulators): *./runFanoutConsumers.sh <serverHost> <clientNum> <progressPrintRateNum>*
   where:
   - *serverHost* - fan-out server's host;
   - *clientNum* - number of consumer clients;
   - *progressPrintRateNum* - the rate of progress indicator in the console, e.g. "Total number of already received Snapshots(W): 45000". Optional: default value is every 1000 msgs;

```
$> ./runFanoutConsumers.sh perf1 200
Press Enter to shutdown
```

4. Run producer clients (Exchange simulators): *./runFanoutProducer.sh <serverHost> <clientNum> <msgPerSession> <msgPerSecond> <fanoutMsgNum>*
   where:

   - *serverHost* - fan-out server's host;
   - *clientNum* - the number of producer clients (Exchange FIX sessions);
   - *msgPerSession* - total number of messages to send per FIX session;
   - *msgPerSecond* - the rate of messages per session. E.g. 100 means a message every 10msec;
   - *fanoutMsgNum* - fanout setting. E.g. 50 means mark each 50th as expected to send in fan-out scenario; Optional: by default equals to clientNum;

   ```
   $> ./runFanoutProducers.sh perf1 50 100000 100
   [INFO ][SLF4JLogFactory]: SLF4J logger wrapper initialized
   [INFO ][ClassPathResourceLoader]: Load resource:/home/bmtest/fixaj/benchmarks/etc/benchmark/fanout/client
   /producer/fixengine.properties
   ...
   ```

   Producer clients program uses a test message template (*benchmarks/etc/benchmark/fanout/producer/snapshot*.fix) to send it to the fan-out server. It needs to wait till it sends all messages to the server, stop its works and exit.

5. Producer clients program exits when it send all messages. Then it needs to stop manually Consumer clients program and fan-out server:

   ```
   ^C
   4999849 packets captured
   9999698 packets received by filter
   0 packets dropped by kernel
   $>
   ```

   Note: Please check that no dropped packets on fan-out server. Otherwiseplease check your environment.

6. Run performance report generation: *./runFanoutReporter.sh <pcapFilePath> <skipNum>*
   where:

   - *pcapFilePath* - path to pcap file to parse and build report;
   - *skipNum* - number of first records which should be not added to histogram (warmup period of all fan-out benchmark components); Optional: default value is 1000;

   ```
   $> ./runFanoutReporter.sh tcpdump.pcap 1000
   IN rate: 4955 (msg/sec), OUT rate: 19840 (msg/sec)
   MIN: 17 (us), MAX: 49151 (us), MEAN: 152.79 (us)
   25.00 percentile: 79 (us)
   50.00 percentile: 135 (us)
   75.00 percentile: 183 (us)
   90.00 percentile: 287 (us)
   99.00 percentile: 607 (us)
   99.90 percentile: 863 (us)
   99.99 percentile: 3071 (us)
   ```