Background

- https://forums.foundationdb.org/t/how-do-bindings-get-an-appropriate-copy-of-fdb-c/311
- https://forums.foundationdb.org/t/new-api-to-get-the-version-of-a-running-cluster-without-needing-a-matching-libfdb-c-library/437

New fdb c API to ask a cluster its protocol version

Proposed API:

```
// Get the server's protocol version, optionally passing in an expected protocol version.
// If you only want to know what the current server protocol is, then pass NULL for currentProtocol.
// Otherwise, the returned future becomes ready when the protocol version different from *currentProtocol.
FDBFuture* fdb_get_server_protocol(const char* clusterFilePath, const uint64_t* currentProtocol);
fdb_error_t fdb_future_get_uint64( FDBFuture* f, uint64_t* out );
```

Basically you can get a future where if it becomes ready, then a majority of coordinators are/were running fdbservers with this protocol version. This will work even with older versions of fdbserver before we added this feature, since they already share their protocol version.

This can be combined with the following existing (but currently undocumented) feature to get the bare minimum answer to a popular question - "Is my client protocol-compatible with a majority of the coordinators it's trying to connect to?"

```
DLLEXPORT const char* fdb_get_client_version();
```

Which returns version information formatted like `$MAJOR.$MINOR.$PATCH,$GIT_HASH,$HEX_PROTOCOL_VERSION`.

We could conceivably also have `fdb_get_server_protocol` return all the same version information that `fdb_get_client_version` does, but this can't be done in a way that will work with old fdbservers since they just don't share that information.

Implementation

- https://github.com/apple/foundationdb/pull/3858
- https://github.com/apple/foundationdb/pull/4028

The road to getting `npm install --save foundationdb` (and similar) to just work

The current proposal is that we start offering "fat bindings" (similar to the fat jar) which bundle a copy of libfdb_c for every supported platform. Such

a package would then be set up to query the server for its protocol version, and if necessary download an appropriate libfdb_c based on that. This downloaded libfdb_c could then be loaded as an external client by the multiversion client. Loading an external client after setting up the network isn't currently supported, so we'd have to fix that.

Retrieving `libfdb_c.so`

We currently have two ideas for retrieving the libfdb_c.so file.

1. Downloading directly from coordinators
2. Downloading from blob store

Both ideas have some considerations that go along with them.

Downloading from coordinators would require thought into where to write the file and how to get write permissions into that directory. We would also have to make sure not to overload coordinators with download requests.

Downloading from blob store would require clients to go to an external source (namely the blob store) to download the libfdb_c file, which some clients may not accept.

Other ideas

- Download libfdb_c directly from coordinators
- Where to write the file? How to get permissions?
- Distribute libfdb_c to all clients before upgrade
- Avoid performance penalty for going through two network threads
- Rules for preferred client when there are multiple protocol-compatible clients
- Don't ping coordinators with all clients
- Push patch updates to clients through server - something like `fdbcli --exec 'distribute client /path/to/libfdb_c.so'`
- Don't overload coordinators
- Measure client downtime caused by recovery, and don't make it worse
- Implement fdb_get_server_protocol natively in clients?