Groups improvements

See also:

- Group contacts management.
- Create groups without establishing direct connections.

Problem

Establishing connections in groups is unstable and uses a lot of traffic. There are several areas for improvement that that could help optimize it:

- Joining group member prematurely creates direct and group connections for each member.

  Some members may never come online, and that traffic would be completely wasted.

  Instead of creating direct connections, we could allow to send direct messages inside group, and optionally have a separate protocol for automating establishing direct connection with member via them.

- Host sends N introduction messages (XGrpMemIntro) to joining member. Instead they could be batched.

Possible solutions

Improved group handshake protocol

Below are proposed changes to group handshake protocol to reduce traffic and improve stability.

Each joining member creates a new temporary per group address for introduced members to connect via. Joining member sends it to host when accepting group invitation.

```
haskell XGrpAcptAddress :: MemberId -> ConnReqContact ->
ChatMsgEvent 'Json
```

Host sends group introductions in batches, batching smaller messages first (introductions of members without profile picture).

For each received batch of N introductions joining member creates N transient per member identifiers (MemberCodes) and replies to host with batched XGrpMemInv messages including these identifiers. Joining member would then use them to verify contact requests from introduced members.

How is MemberCode different from MemberId? - MemberId is known to all group members and is constant per member per group. MemberCode would

be known only to host and to introduced member (of existing members), so other members wouldn't be able to impersonate one another when requesting connection with joining member. An introduced member can still pass their identifier + joining member address to another member or outside of group, but it is no different to passing currently shared invitation links.

```haskell newtype MemberCode = MemberCode {unMemberCode :: ByteString}

XGrpMemInvCode :: MemberId -> MemberCode -> ChatMsgEvent 'Json

-- instead of / in addition to batching message could be

type MemberCodes = Map MemberId MemberCode

XGrpMemInvCodes :: MemberCodes -> ChatMsgEvent 'Json ```

Host includes joining member address and code (unique for each introduced member) into XGrpMemFwd messages instead of invitation links:

```haskell XGrpMemFwdCode :: MemberInfo -> ConnReqContact -> MemberCode -> ChatMsgEvent 'Json```

Introduced members send contact requests with a new message XGroupMember / XIntroduced (similar to XInfo or XContact, see `processUserContactRequest`):

```haskell XIntroduced :: MemberInfo -> MemberCode -> ChatMsgEvent 'Json```

Joinee verifies profile and code and automatically accepts contact request. They both assign resulting connection to respective group member record, without creating contact.

After (if) all introduced members have connected, joining member deletes per group address. Possibly it can also be deleted after expiration interval.

**Group links**

We can reduce number of steps taken to join group via group link:

- Do not create direct connection and contact with group link host, instead use the connection resulting from contact request as a group connection, and assign it to a group member record.
- Host to not send XGrpInv message, joining member to not wait for it, instead joining member would initiate with XGrpAcptAddress after establishing connection via group link.

In addition to their profile, host includes MemberId of joining member into confirmation when accepting group link join request, using new message:
```

```
haskell XGroupLinkInfo :: Profile -> MemberId -> ChatMsgEvent
'Json
```

Joining member initially doesn't know group profile, they create a placeholder group with a new dummy profile (alternatively, we could include at least group display name into group link). After connection is established, host sends XGrpInfo containing group profile to joining member. This can happen in parallel with group handshake started by XGrpAcptAddress.

Group profile could also be included into XGroupLinkInfo if not for the limitation on size if both host's profile and group profile contain pictures.

Adding member to the group

Clients compatibility

We have a proposed mechanism for communicating "chat protocol version" between clients.

Sending and processing new protocol messages would only be supported by updated clients.

Trying to support both protocols across different members in the same group would require complex logic:

Host would have to send introduced members versions, joining member would provide both address or invitation links depending on each members' versions, host would forward accordingly.

Instead we could assign "chat protocol version" per group and share it with members as part of group profile, and make a two-stage release when members would first be able to update and get new processing logic, but have it disabled until next release.

After group switching to new processing logic old clients wouldn't be able to connect in groups.

How should existing groups be switched?

- Owner user action?
- Owner client deciding automatically?
- In case group has multiple owners - which owner(s) can / should decide?
- Prohibited until all / part of existing members don't update? How to request members to update?
- Old clients will not be able to process and save group chat version from group profile update.

Sending direct messages inside group

Group messages are sent by broadcasting them to all group member connections. As a replacement for creating additional direct connections in group we can allow to send message directly to members via group member

connections. The UX would be to choose whether to send to group or to a specific member via compose view.

Possible approach is to extend ExtMsgContent with `direct :: Maybe Bool` field, which would only be considered for group messages.

Chat items should store information of receiving member database ID (for sending member) and of message being direct (for receiving member). Perhaps it could be a single field `direct_member_id`, which would be the same as `group_member_id` for received messages.

TODO - consider whether `connection_id` or `group_id` or both should be assigned in `messages` table.