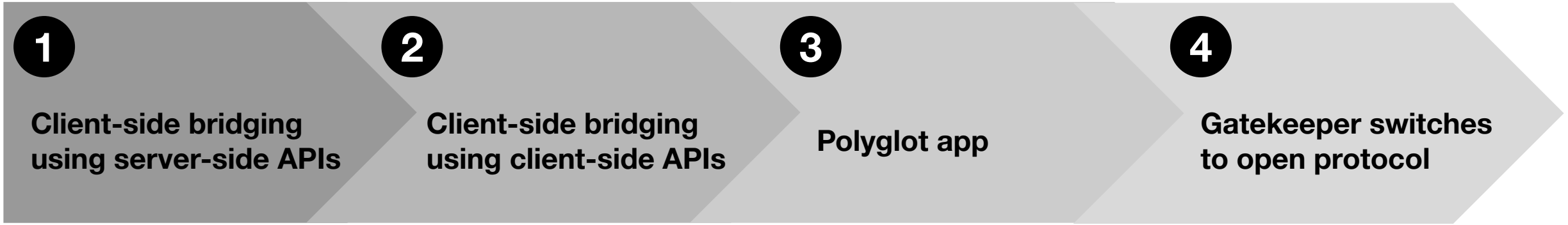
Implementing Interoperability for the DMA

matthew@matrix.org | amandine@matrix.org
@matthew:matrix.org | @Amandine:matrix.org

Interoperability Implementation Options



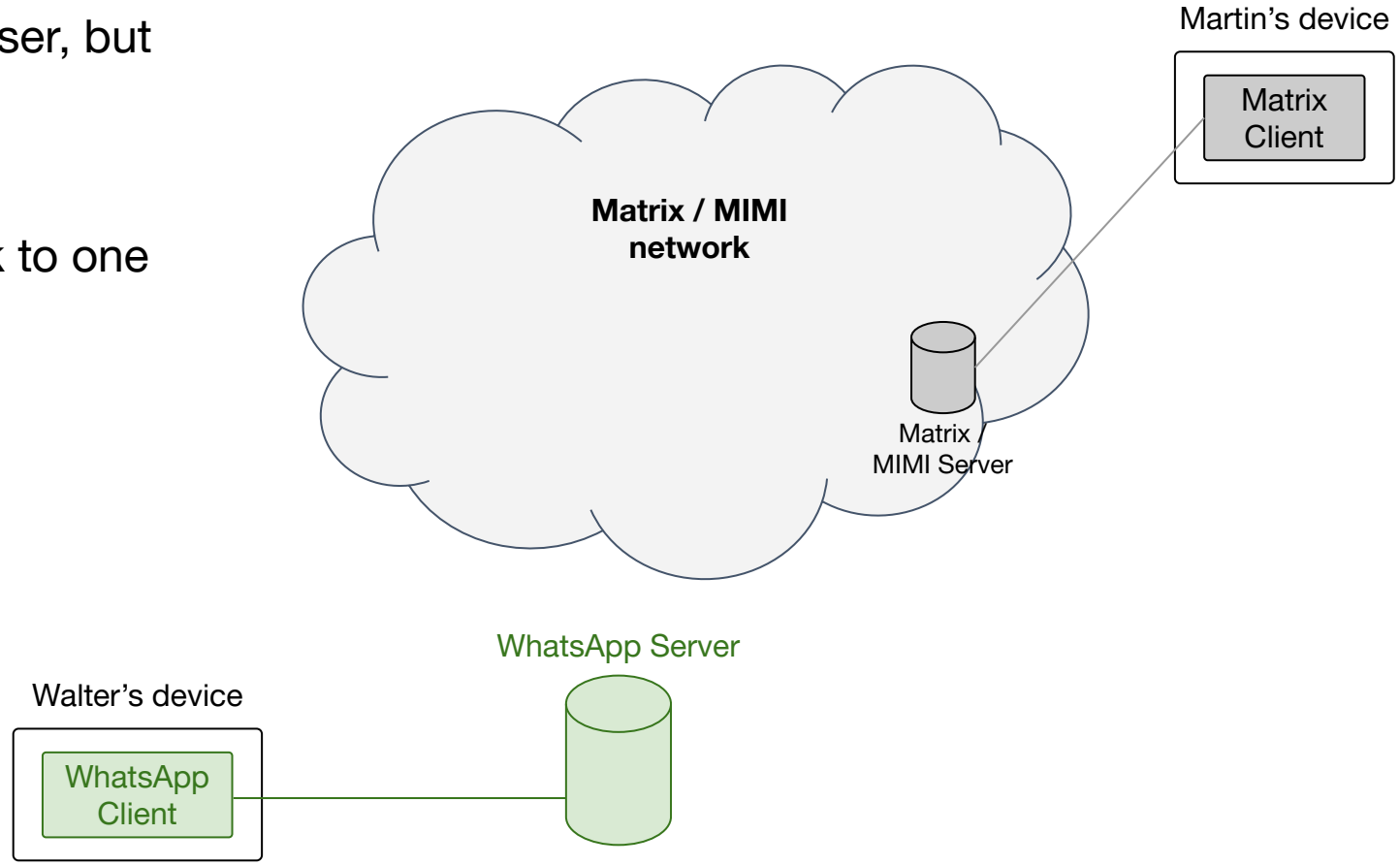
- These options can be implemented in a standalone manner or seen as a transition towards a full implementation of an open protocol.
- In this presentation we made the choice of using an open protocol (it could be Matrix, MIMI, XMPP, RCS, etc…); but all these options are valid with any protocol, including closed ones.*

* Not using an open protocol only means that if one player requests interoperability with several gatekeepers, they will have to implement the chosen interoperability option for each of the gatekeeper's protocols they want to interoperate with, and precluding option 4.

Problem to be solved

- **Walter** is a WhatsApp (gatekeeper, GK) user and doesn't care about interop.
- **Martin** is a Matrix (requesting party, RP) user, but doesn't want to install WhatsApp.

We want Walter and Martin to be able to speak to one another.



1 Client-side bridging using server-side APIs

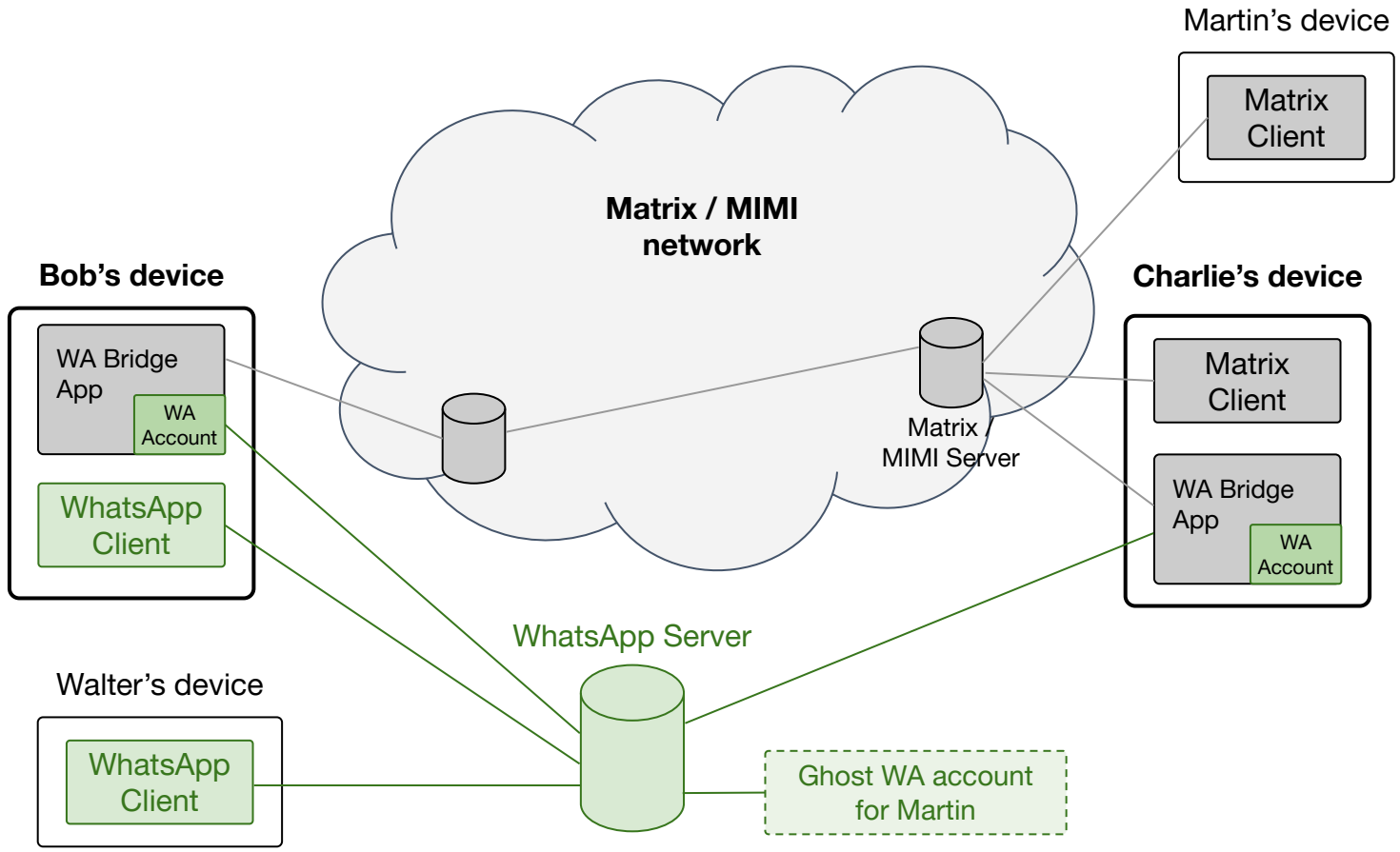
matrix

- **Bob** is a WhatsApp user and wants to talk to people who are not on WhatsApp:

  ➡ Install a bridge app to expose his WhatsApp account to other users.

- **Charlie** has a WhatsApp account but doesn't want to use the WhatsApp app.

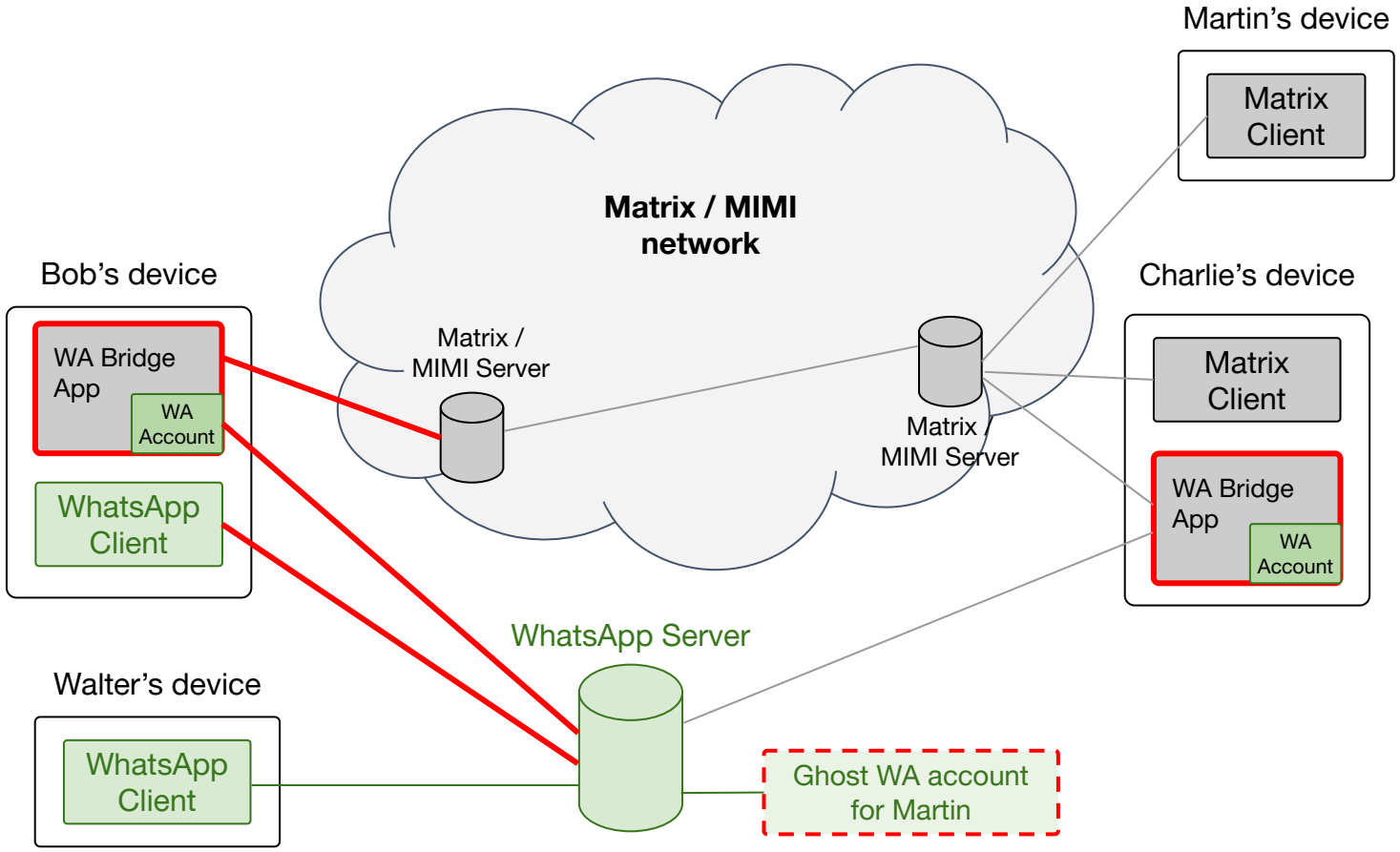
  ➡ Install a bridge app to connect his interoperable client to his WA account.



1 Client-side bridging using server-side APIs

matrix

What it means for the gatekeeper (GK)	• The GK needs to expose the server-side APIs allowing the RP to develop a bridge • The GK needs a way to represent remote users • Simplest / quickest to implement, pragmatic but ugly
What it means for the requesting party (RP)	• The RP needs to source and advertise a bridge
What it means for the users	• GK's users can install the bridge to expose themselves to the RP's users • RP's users can install the bridge to connect to GK's users (need an account on the GK's network)

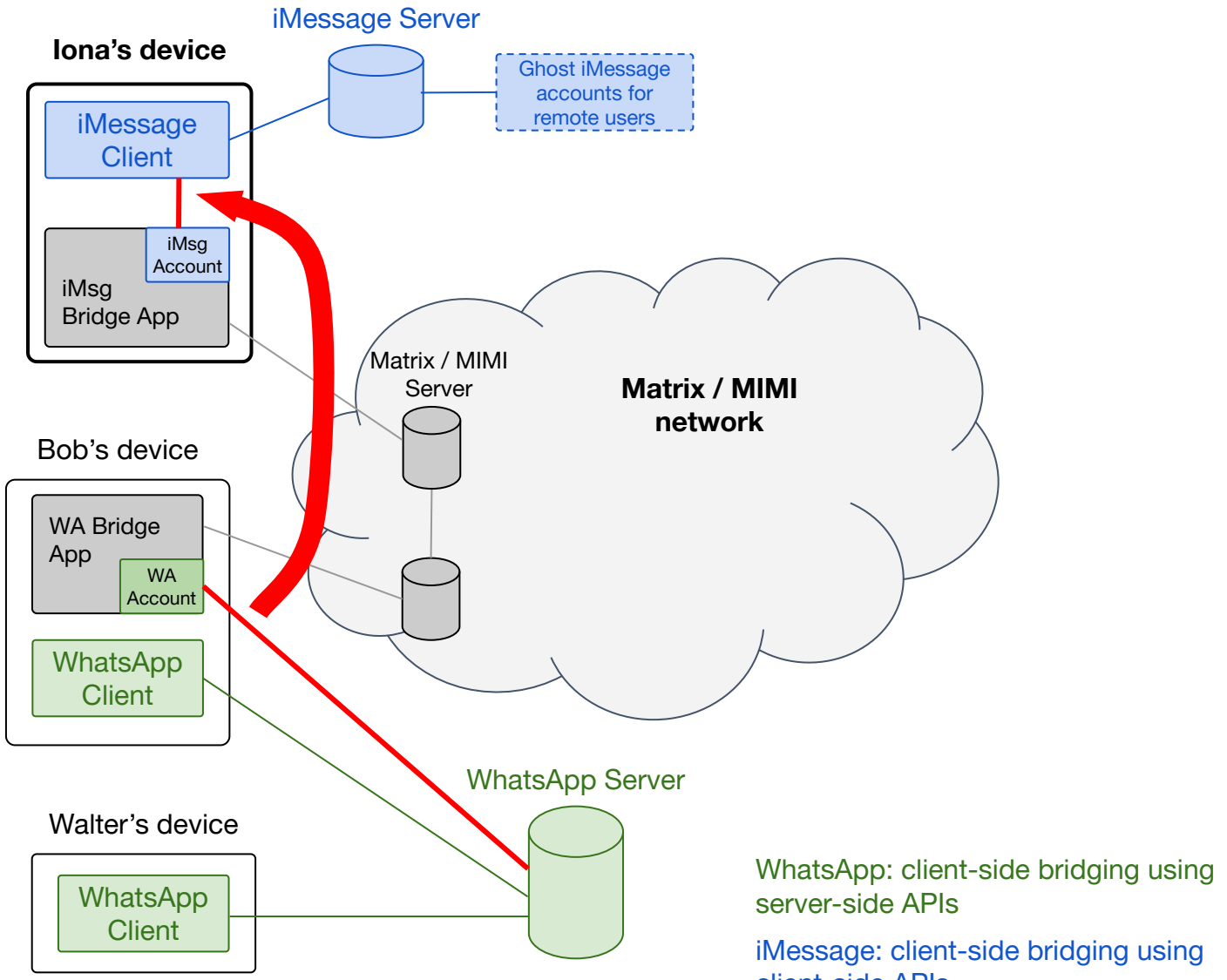


2 Client-side bridging using client-side APIs

matrix

- **Iona** is an iMessage user who wants to communicate with non-iMessage users, or use another app to / user interface to communicate.

  ➡ Install a bridging app which connects directly to her local iMessage client.



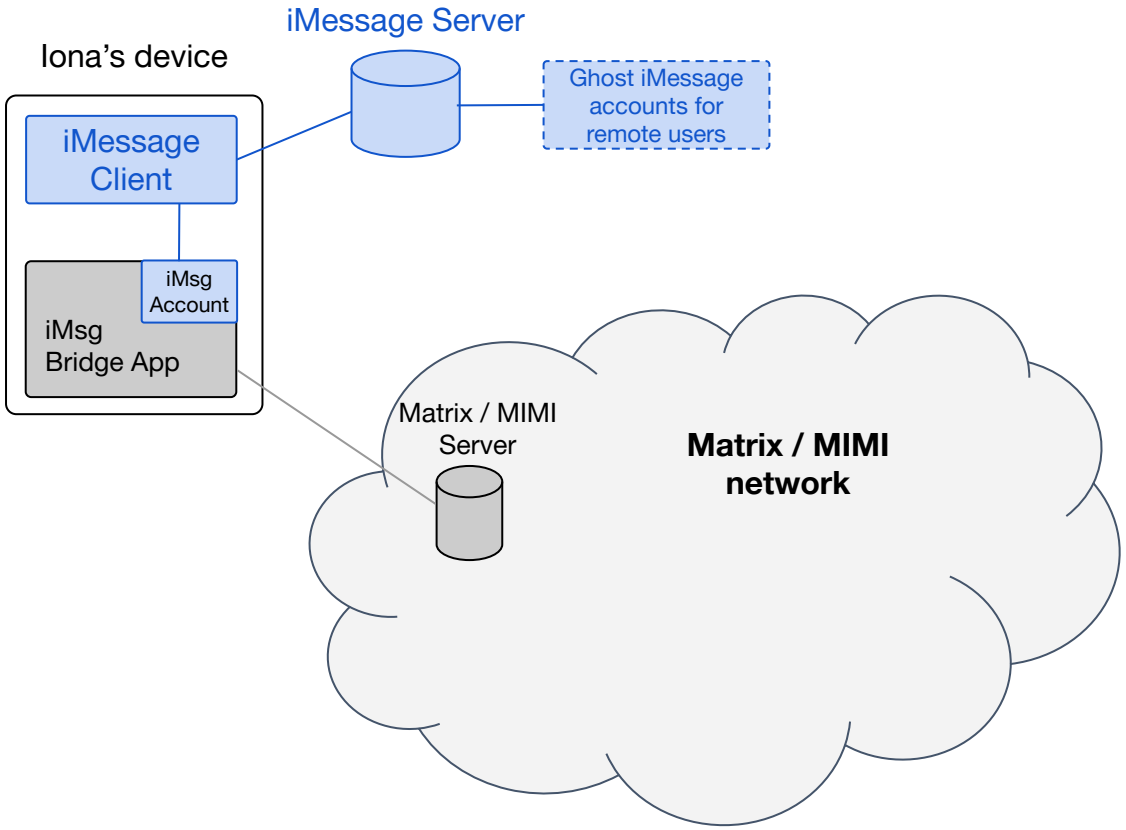
WhatsApp: client-side bridging using server-side APIs

iMessage: client-side bridging using client-side APIs

2 Client-side bridging using client-side APIs

matrix

What it means for the gatekeeper (GK)	• The device's operating system has to provide an interface for the client-side APIs • The GK keeps their app as it is, other than exposing the client-side APIs allowing the RP to develop a bridge • The GK needs a way to represent remote users
What it means for the requesting party (RP)	• The RP needs to source and advertise a bridge
What it means for the users	• GK's users can install the bridge to expose themselves to the RP's users • RP's users can install the bridge to connect to GK's users (need an account on the GK's network)

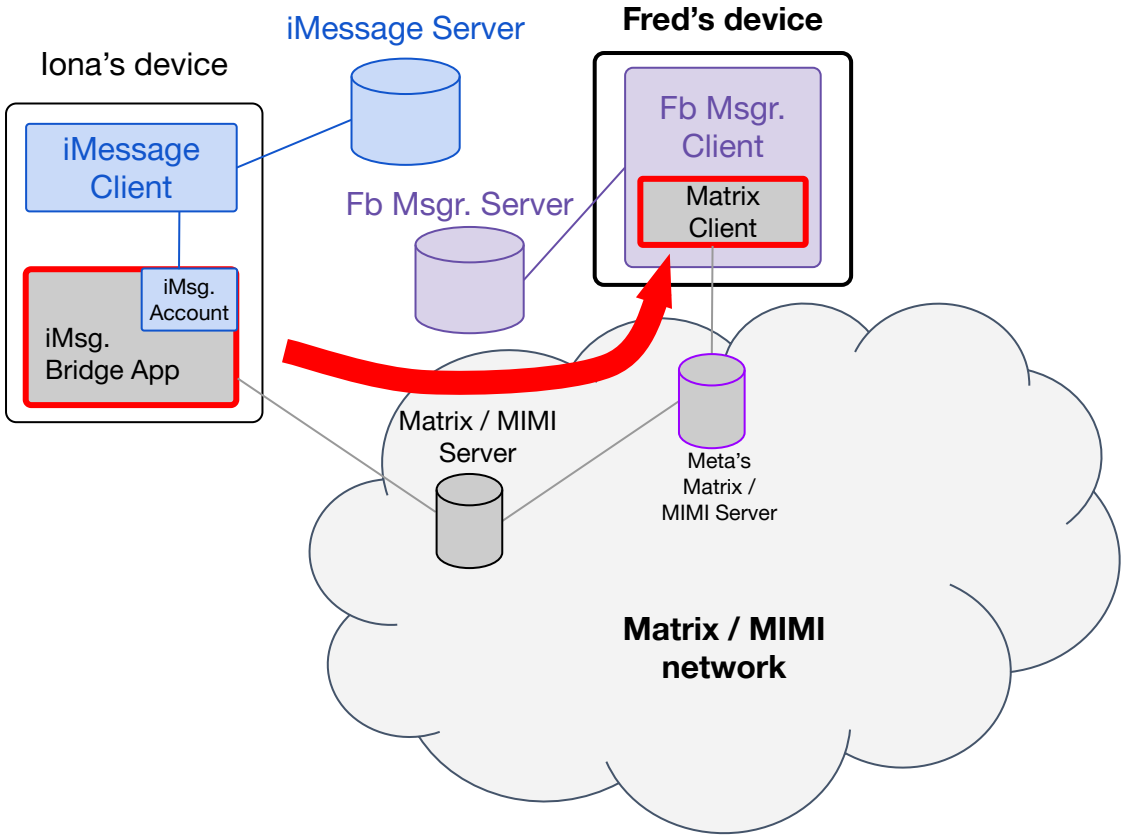


3 **Polyglot app** (Gatekeeper's client adds 3rd party protocol support alongside the legacy service)

matrix

- **Fred** is a Facebook Messenger user.

  ➡ Let's assume Facebook has added Matrix/MIMI/etc alongside the existing FB stack

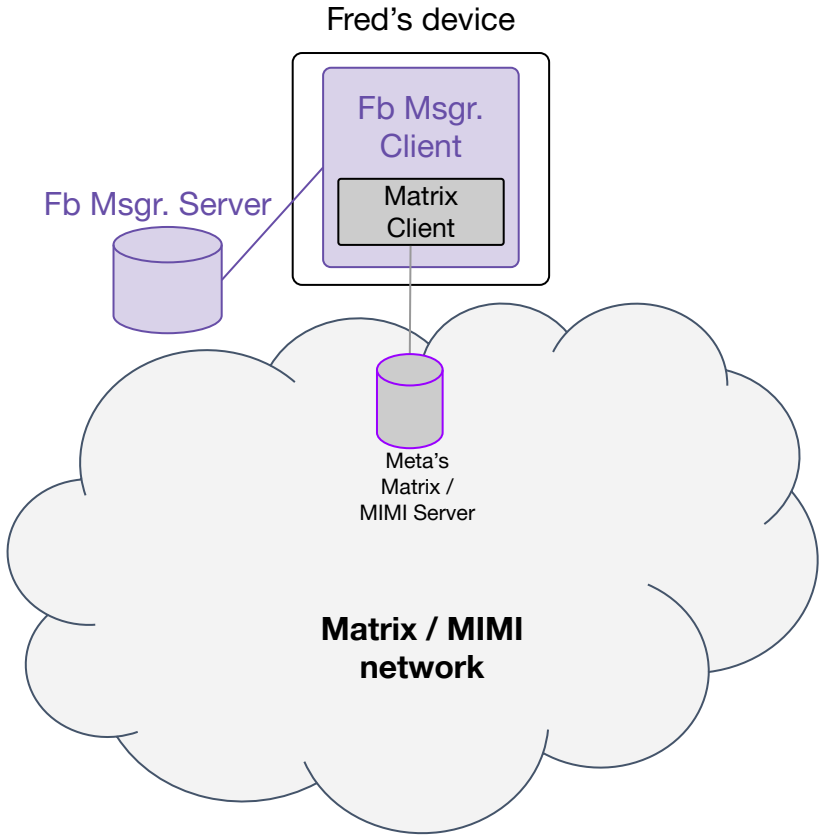


iMessage: client-side bridging using client-side APIs

Facebook Messenger: polyglot app

3 Polyglot app (Gatekeeper's client adds 3rd party protocol support alongside the legacy service)

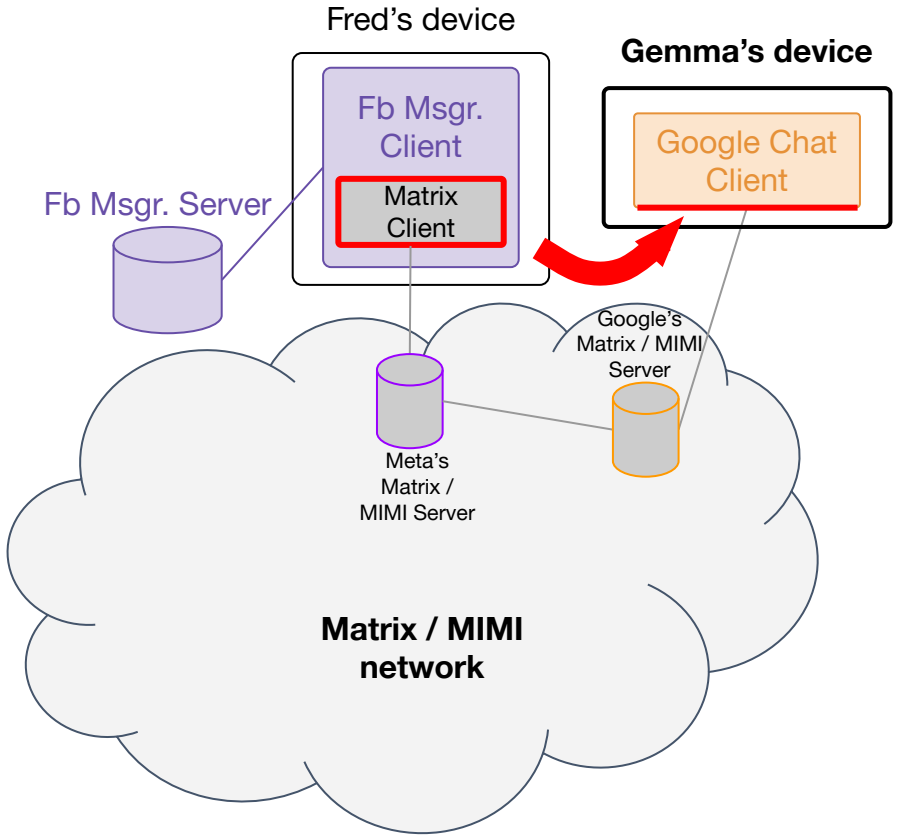
What it means for the gatekeeper (GK)	• The GKs client adds support for a 3rd party protocol as well as speaking to the legacy service • The 3rd party protocol can be an open protocol or the RP's protocol • Heavy implementation if GK implements several 3rd party protocols
What it means for the requesting party (RP)	• The RP needs to implement the protocol chosen by the GK, unless it's their own. • The RP will have to do it for every GK they want to interoperate with → x^n effort
What it means for the users	• RP's users can seamlessly talk to GK users and vice versa • RP's users may either use their own Matrix ID or the GK may generate one automatically



4 Gatekeeper switches to open protocol

- **Gemma** is a Google Chat user.

  => Let's assume Google Chat decides to switch to talk Matrix/MIMI natively.

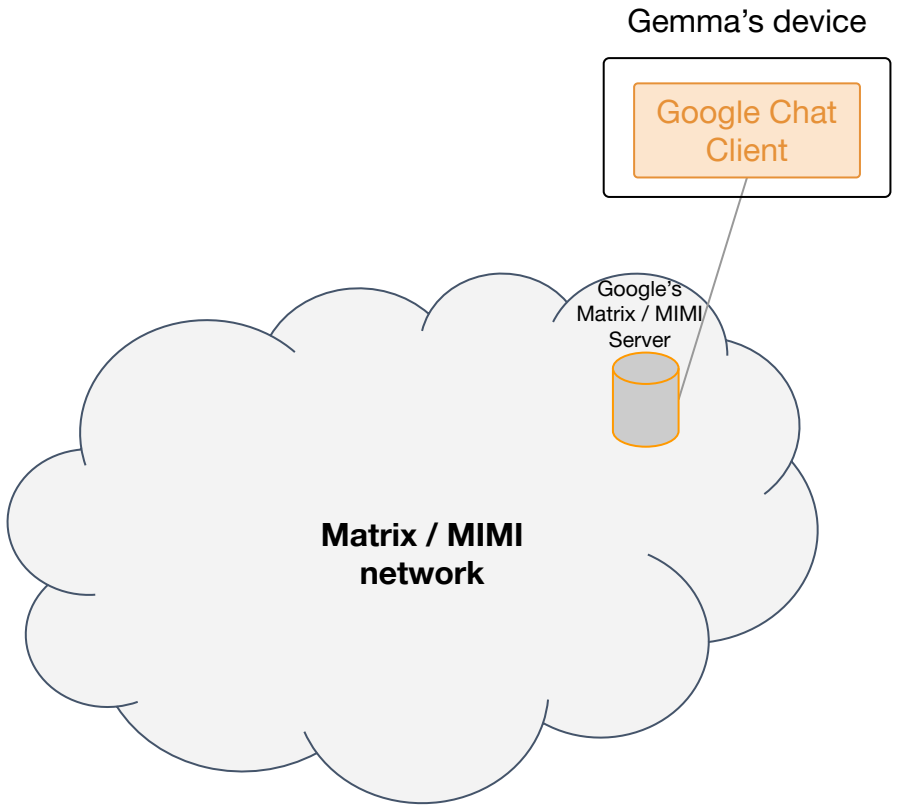


Facebook Messenger: polyglot app

Google Chat: native Matrix/MIMI support

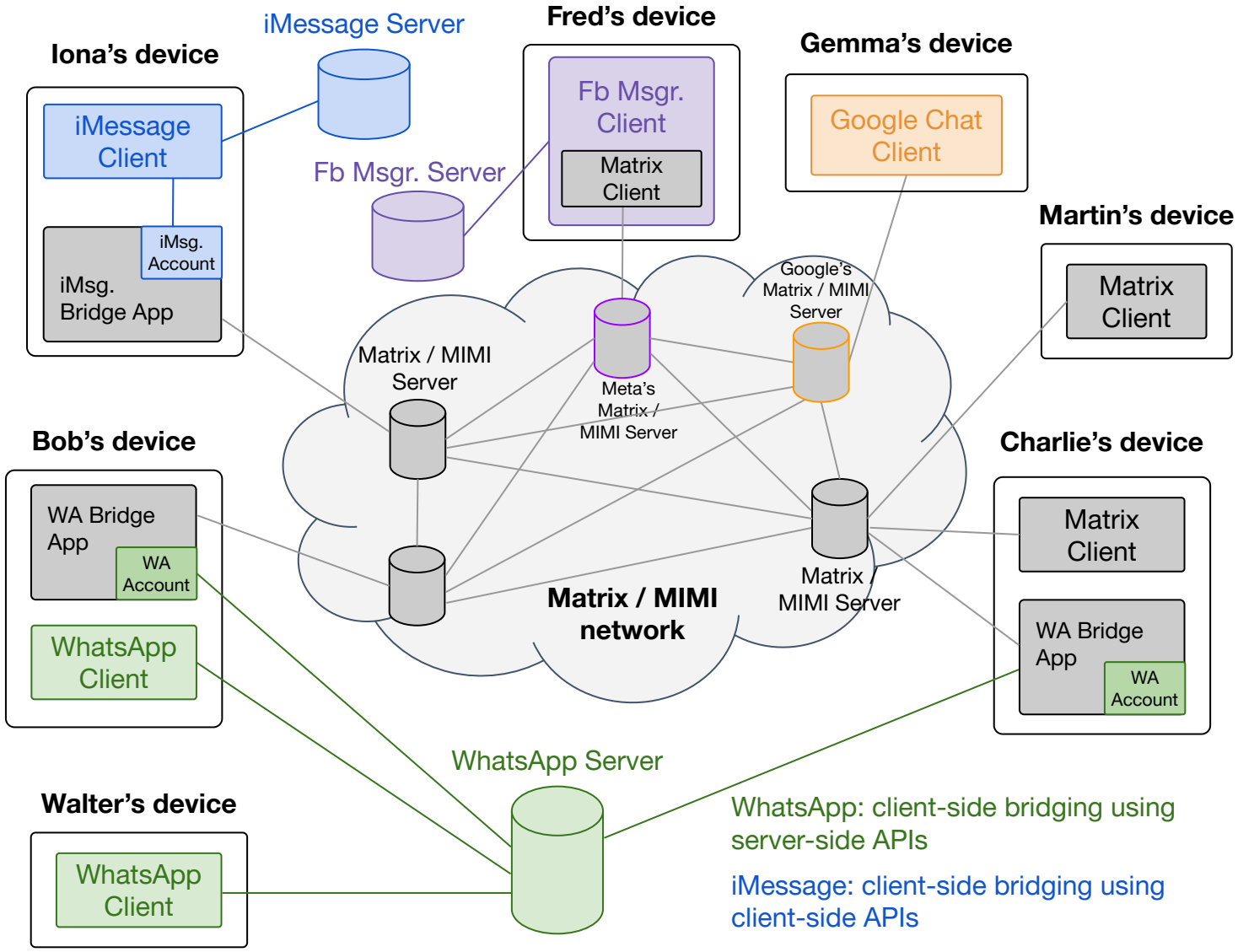
4 Gatekeeper switches to open protocol

What it means for the gatekeeper (GK)	• The GK speaks the 3rd party protocol entirely • The GK can interoperate with anyone implementing the open protocol • More invasive change from an implementation perspective, but more sustainable too and easier to maintain.
What it means for the requesting party (RP)	• The RP needs to implement the open protocol chosen by the GK • It will enable them to interoperate with anyone using the chosen open protocol
What it means for the users	• RP's users can seamlessly talk to GK users and vice versa



Interoperability is implementation agnostic

- Walter is a WhatsApp user and doesn't care about interop.
- Bob is a WhatsApp user and wants to talk to people who are not on WhatsApp.
- Charlie has a WhatsApp account but doesn't want to use the WhatsApp app.
- Martin is a Matrix user.
- Iona is an iMessage user who wants to expose herself to non-iMessage users, or use another app to / user interface to communicate.
- Fred is a Facebook Messenger user.
- Gemma is a Google Chat user.



WhatsApp: client-side bridging using server-side APIs

iMessage: client-side bridging using client-side APIs

Facebook Messenger: polyglot app

Google Chat: native Matrix/MIMI support

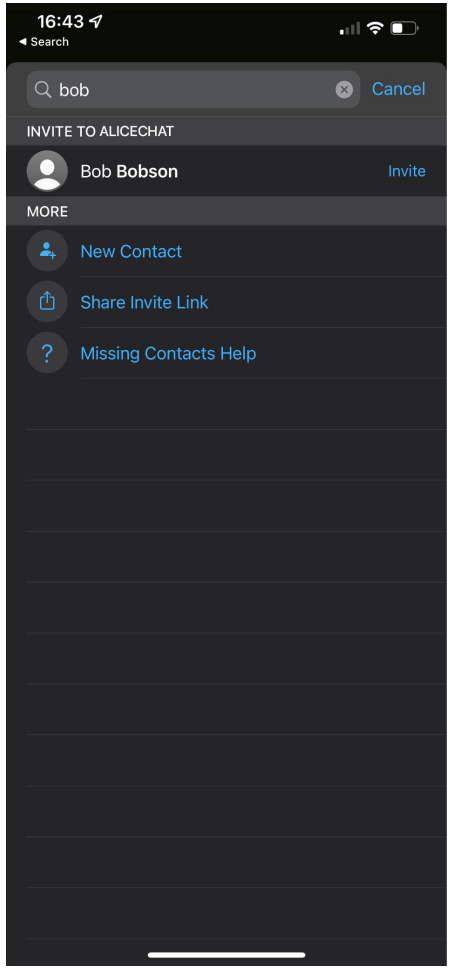
Pros & Cons

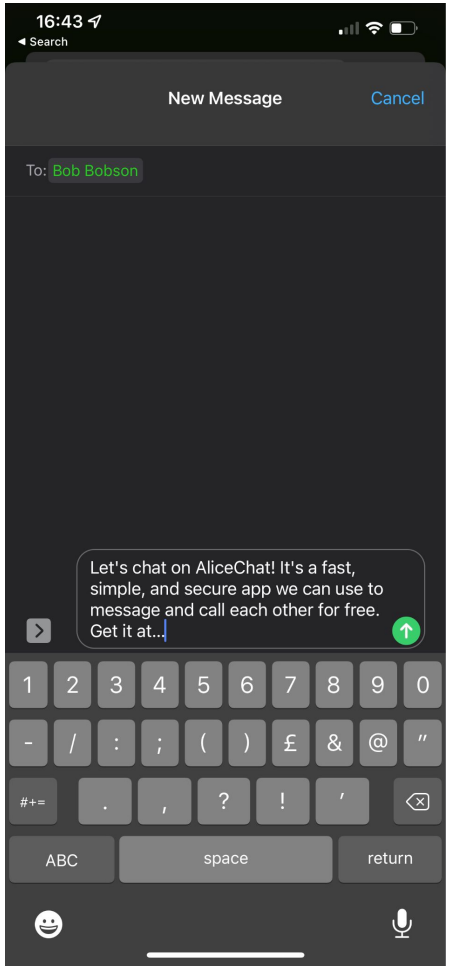
	Pros	Cons
1 - Client-side bridging using server-side APIs *Existing example: Matrix WhatsApp<->GChat demo*	• Minimal effort from GK: just need to document their current API and represent remote users. • Bridges can be supplied by anyone.	• Traffic does 3 client↔server traversals (GK-app → GK-server → bridge-client → RP-server) • GK users have to opt in by installing a separate bridging app.
2 - Client-side bridging using client-side APIs	• GK has to make their app expose a simple client-side API that lets other apps send/recv msgs etc. • Traffic does 1 client↔server traversal (GK-app → bridge-client → RP-server) • Bridges can be supplied by anyone.	• GK users have to opt in by installing a separate bridging app. • OS vendor might need to get involved to define a standard 'intent' or API to let bridge apps talk to GK apps
3 - Polyglot app *Existing example: iMessage's support for SMS*	• GK users can automatically speak to RPs without installing a separate app! • No bridges; the bridge is effectively now within the GK app.	• GK has to add an RP stack to their app alongside their existing stack. • Risk of substandard RP implementation from GK undermines DMA; users might end up using bridge apps anyway.
4 - Gatekeeper switches to open protocol *Existing example: Reddit's implementation of Matrix*	• GK users can automatically speak to all RPs who implement the open protocol • GK can build differentiating features on the open protocol to add value, rather than relying on network effects.	• GK's investment in existing stack is lost.

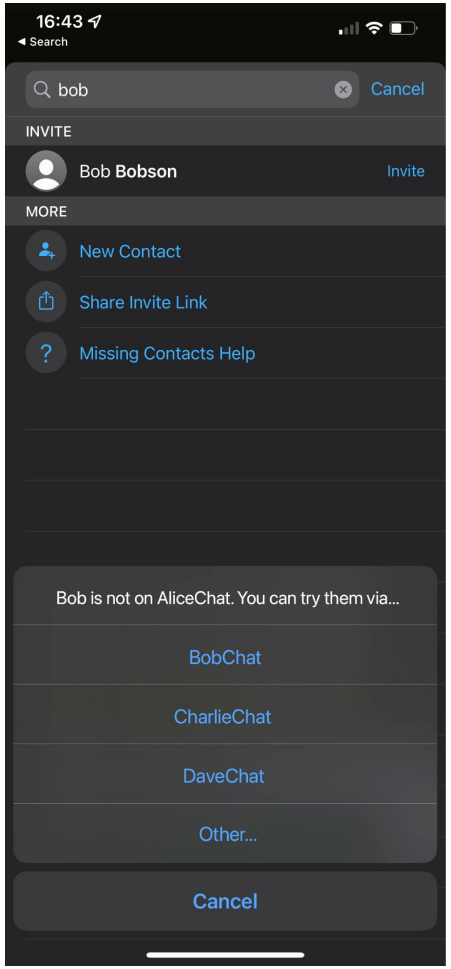
User Discovery
Finding Bob

We've focused on interoperability rather than identity and discoverability, but here is a high level example of how it could work:

- Walter is on WalterChat, an existing gatekeeper
  - E2EE messaging service
  - identifies users using phone numbers
- Walter wants to start a 1-to-1 conversation with Bob
  - Bob doesn't use WalterChat
  - Walter knows he is a keen user of BobChat.







matrix

Thank you

Interoperability Implementation Options

Summary table

Implementation options	What it means for the gatekeeper (GK)	What it means for the requesting party (RP)	What it means for the user
1 - Client-side bridging using server-side APIs *Existing example: Matrix WhatsApp<->GChat demo*	• The GK needs to expose the server-side APIs allowing the RP to develop a bridge • The GK needs a way to represent remote users • Simplest / quickest to implement, pragmatic but ugly	• The RP needs to source and advertise a bridge	• GK's users can install the bridge to expose themselves to the RP's users • RP's users can install the bridge to connect to GK's users (need an account on the GK's network)
2 - Client-side bridging using client-side APIs	• The device's operating system has to provide an interface for the client-side APIs • The GK keeps their app as it is, other than exposing the client-side APIs allowing the RP to develop a bridge • The GK needs a way to represent remote users	• The RP needs to source and advertise a bridge	• GK's users can install the bridge to expose themselves to the RP's users • RP's users can install the bridge to connect to GK's users (need an account on the GK's network)
3 - Polyglot app *Existing example: iMessage's support for SMS*	• The GKs client adds support for a 3rd party protocol as well as speaking to the legacy service • The 3rd party protocol can be an open protocol or the RP's protocol • Heavy implementation if GK implements several 3rd party protocols	• The RP needs to implement the protocol chosen by the GK, unless it's their own. • The RP will have to do it for every GK they want to interoperate with → x^n effort	• RP's users can seamlessly talk to GK users and vice versa • RP's users may either use their own Matrix ID or the GK may generate one automatically
4 - Gatekeeper switches to open protocol *Existing example: Reddit's implementation of Matrix*	• The GK speaks the 3rd party protocol entirely • The GK can interoperate with anyone implementing the open protocol • More invasive change from an implementation perspective, but more sustainable too and easier to maintain.	• The RP needs to implement the open protocol chosen by the GK • It will enable them to interoperate with anyone using the chosen open protocol	• RP's users can seamlessly talk to GK users and vice versa