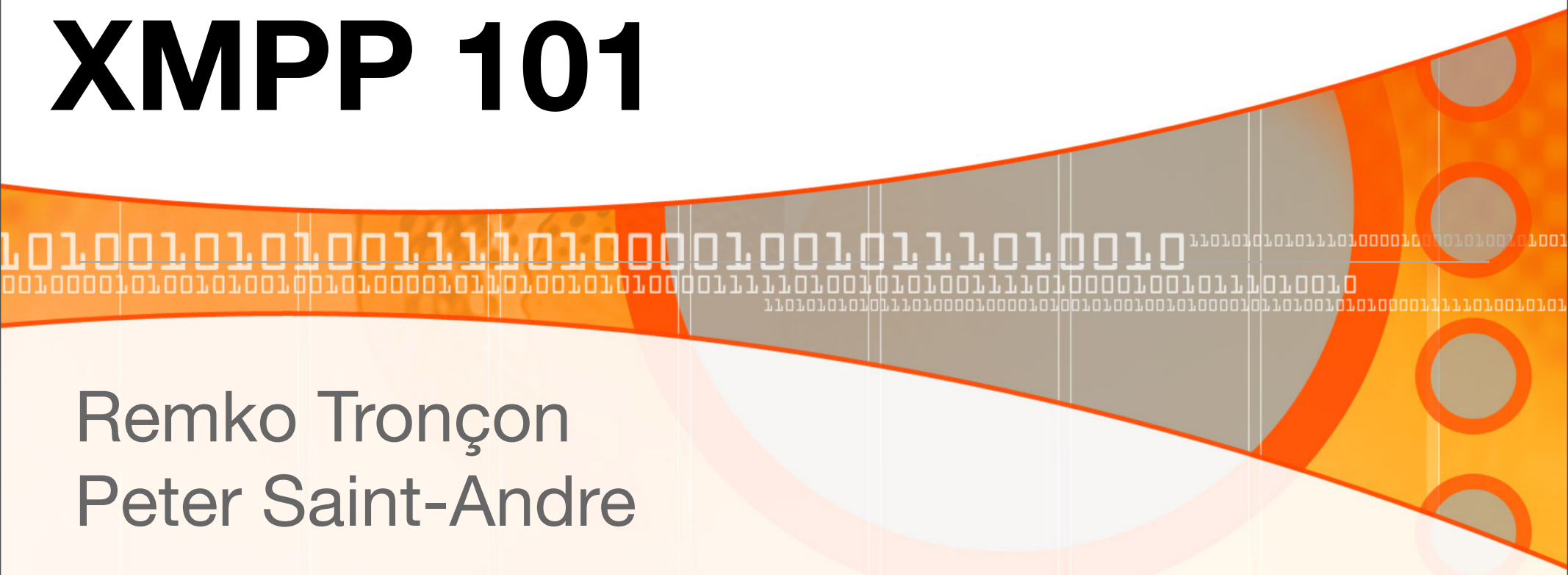
XMPP 101

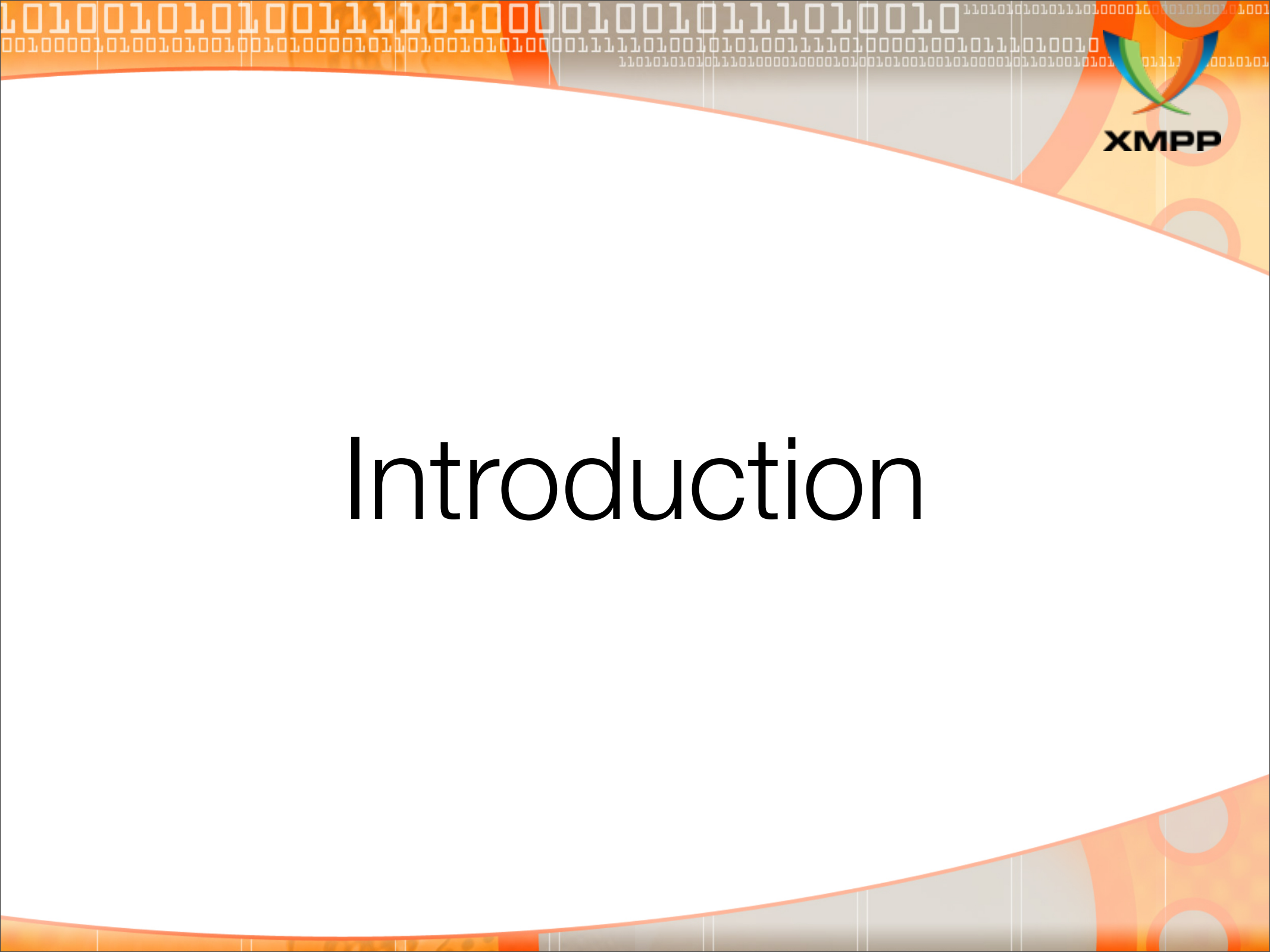


Remko Tronçon
Peter Saint-Andre

XMPP

Overview

- Introduction
- XMPP Basics
- Example code
- Extensions
- State of the bulb
- Conclusion



Introduction

About Us

About Us

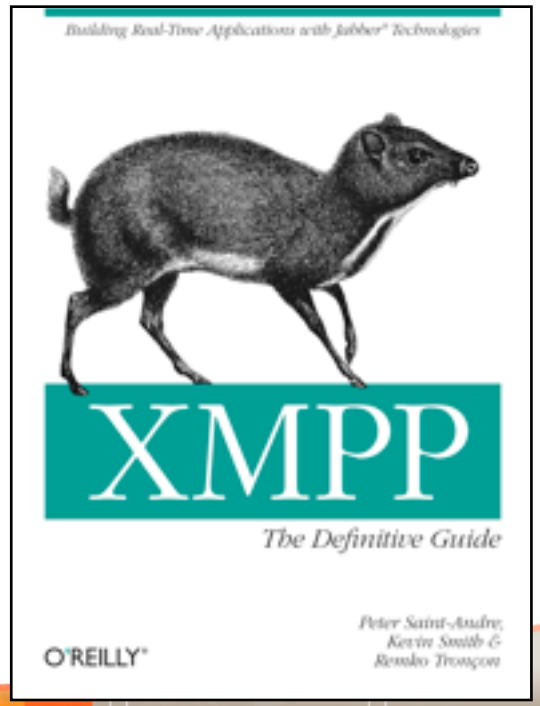
- **Peter**: Documentation guy and specification author

About Us

- **Peter**: Documentation guy and specification author
- **Remko**: Developer for Psi client and other projects

About Us

- **Peter**: Documentation guy and specification author
- **Remko**: Developer for Psi client and other projects
- Co-authors (with Kevin Smith) of *XMPP: The Definitive Guide (O'Reilly, 2009)*



About You

About You

- Why are you here?

About You

- Why are you here?
- What do you want to build?

About You

- Why are you here?
- What do you want to build?
- What is your background? (web, client-server, ...)

What is XMPP?

What is XMPP?

- eXtensible Messaging and Presence Protocol

What is XMPP?

- eXtensible Messaging and Presence Protocol
- Jabber

What is XMPP?

- eXtensible Messaging and Presence Protocol
- Jabber
- Real time messaging system

What is XMPP?

- eXtensible Messaging and Presence Protocol
- Jabber
- Real time messaging system
- Routes small snippets of XML

What is XMPP?

- eXtensible Messaging and Presence Protocol
- Jabber
- Real time messaging system
- Routes small snippets of XML

Alice
alice@wdland.lit

Sister
sister@rlworld.lit

What is XMPP?

- eXtensible Messaging and Presence Protocol
- Jabber
- Real time messaging system
- Routes small snippets of XML

Alice
alice@wdland.lit

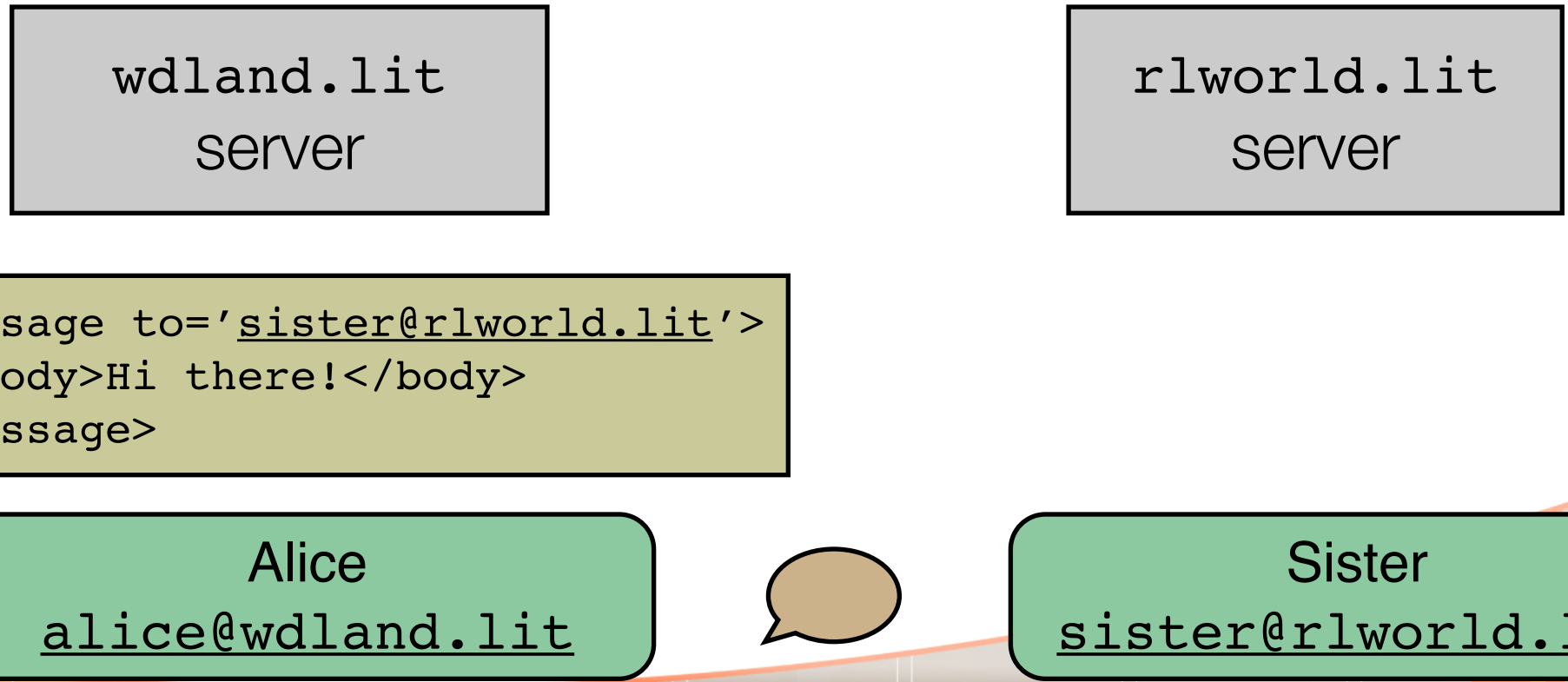
Sister
sister@rlworld.lit

What is XMPP?

- eXtensible Messaging and Presence Protocol
- Jabber
- Real time messaging system
- Routes small snippets of XML

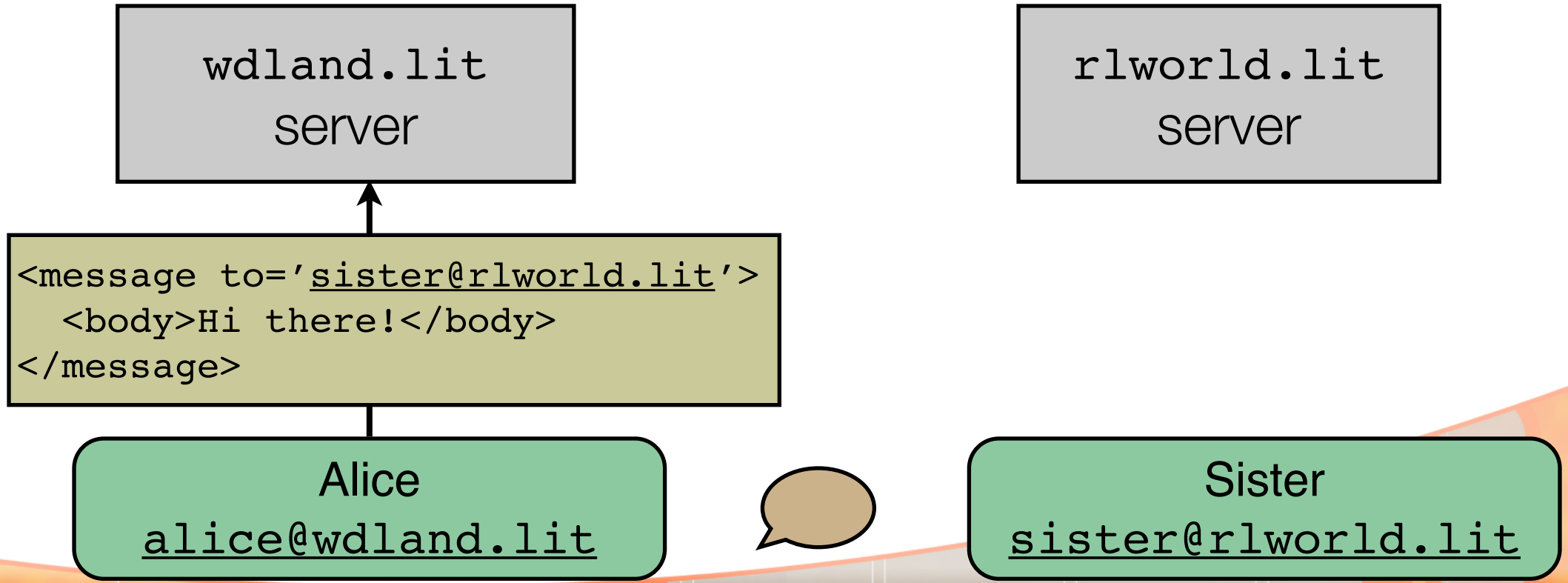
What is XMPP?

- eXtensible Messaging and Presence Protocol
- Jabber
- Real time messaging system
- Routes small snippets of XML



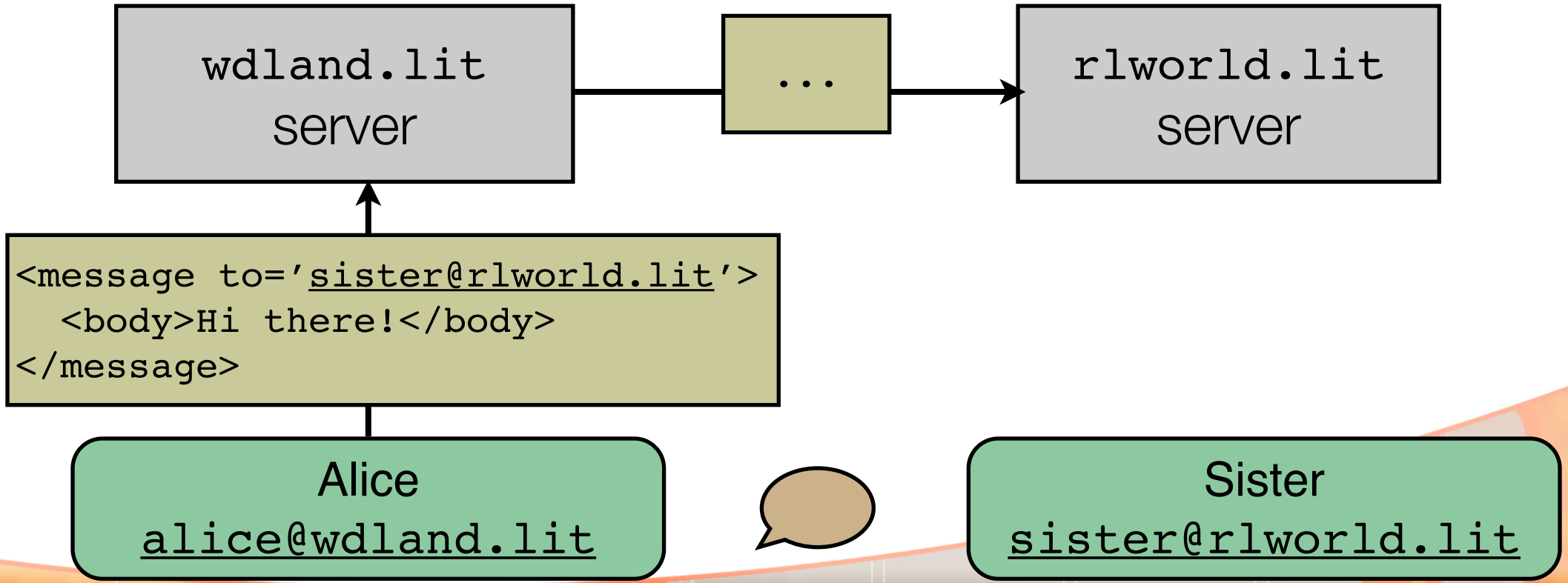
What is XMPP?

- eXtensible Messaging and Presence Protocol
- Jabber
- Real time messaging system
- Routes small snippets of XML



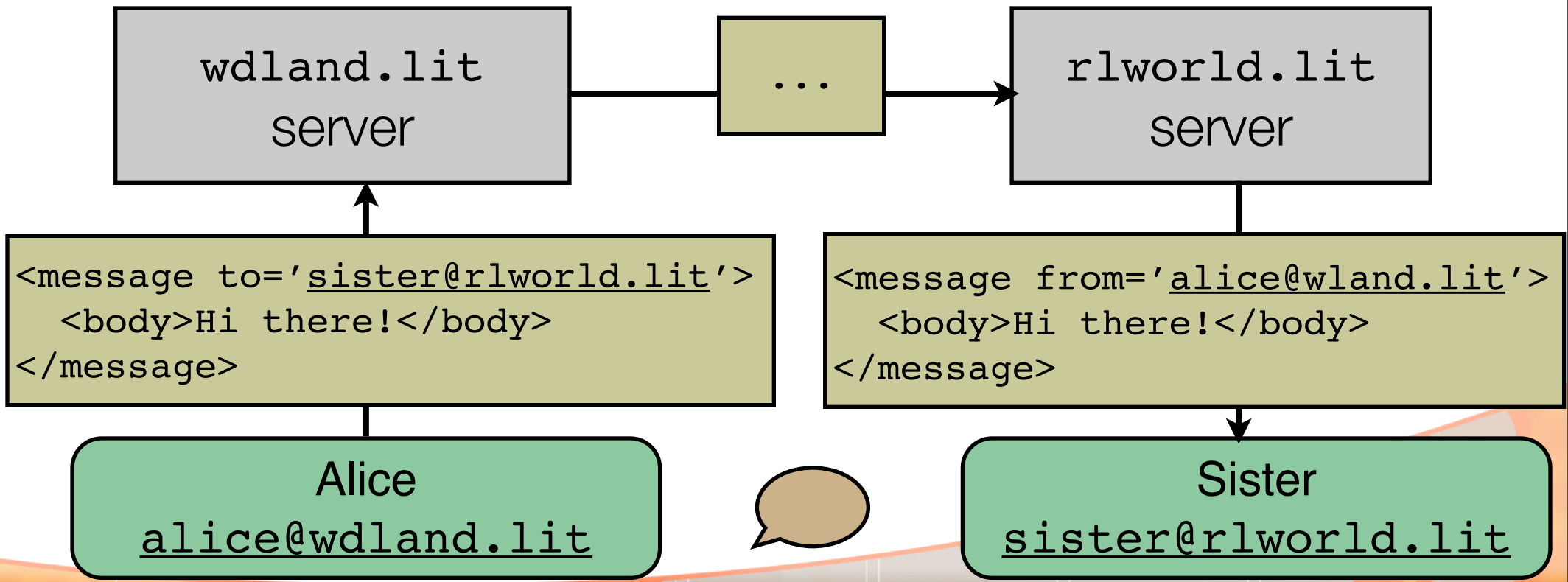
What is XMPP?

- eXtensible Messaging and Presence Protocol
- Jabber
- Real time messaging system
- Routes small snippets of XML



What is XMPP?

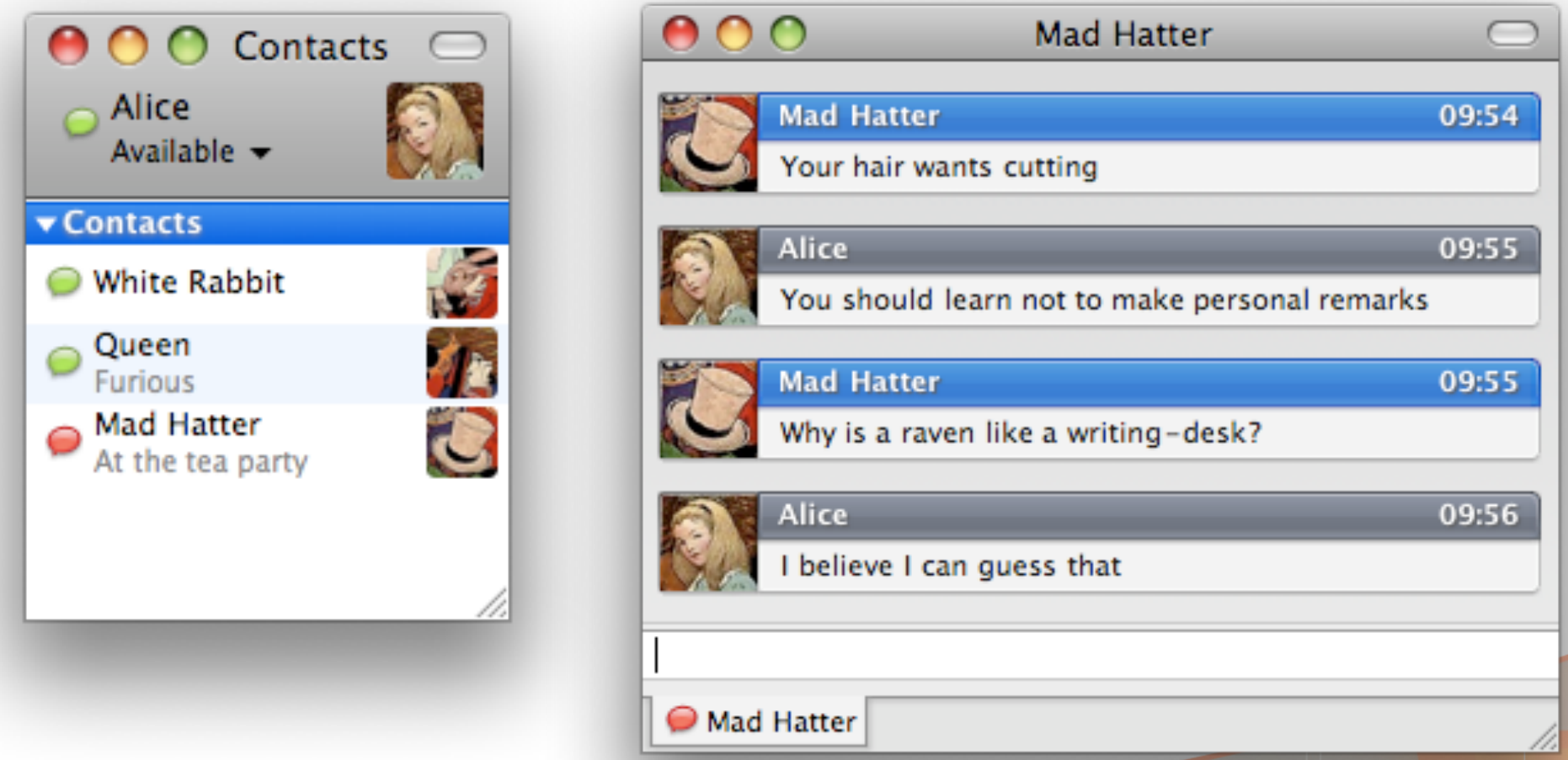
- eXtensible Messaging and Presence Protocol
- Jabber
- Real time messaging system
- Routes small snippets of XML



What can you do with XMPP?

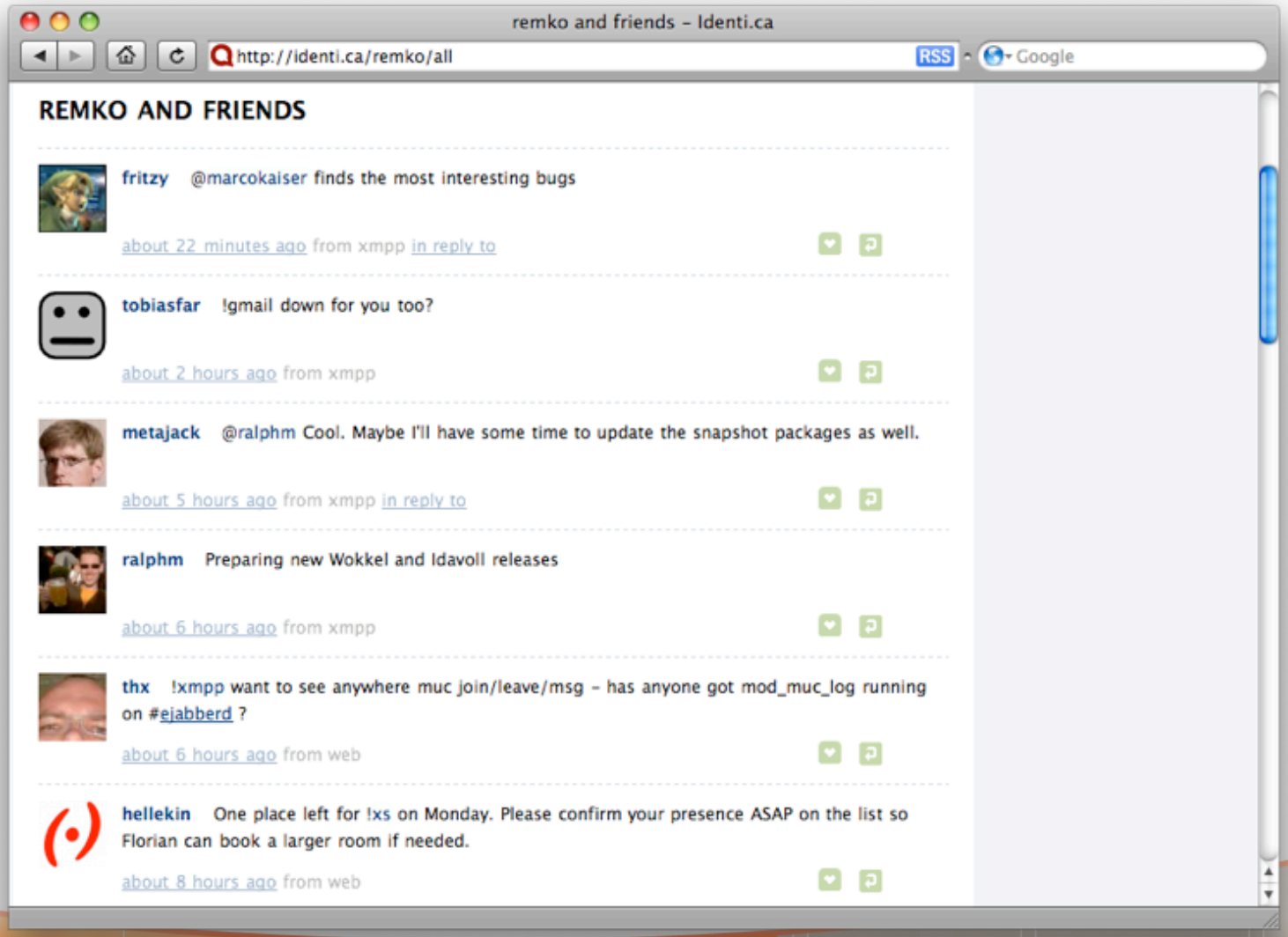
What can you do with XMPP?

Instant Messaging



What can you do with XMPP?

Real-time Social Networking

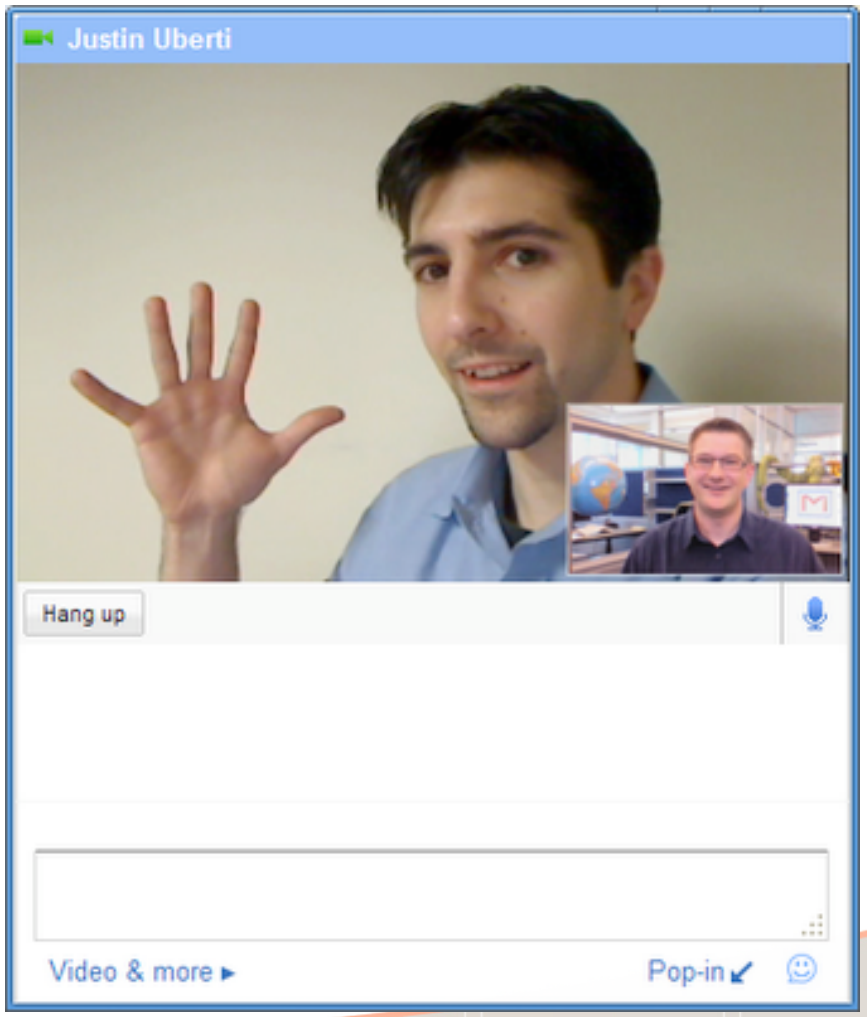


What can you do with XMPP?

Gaming

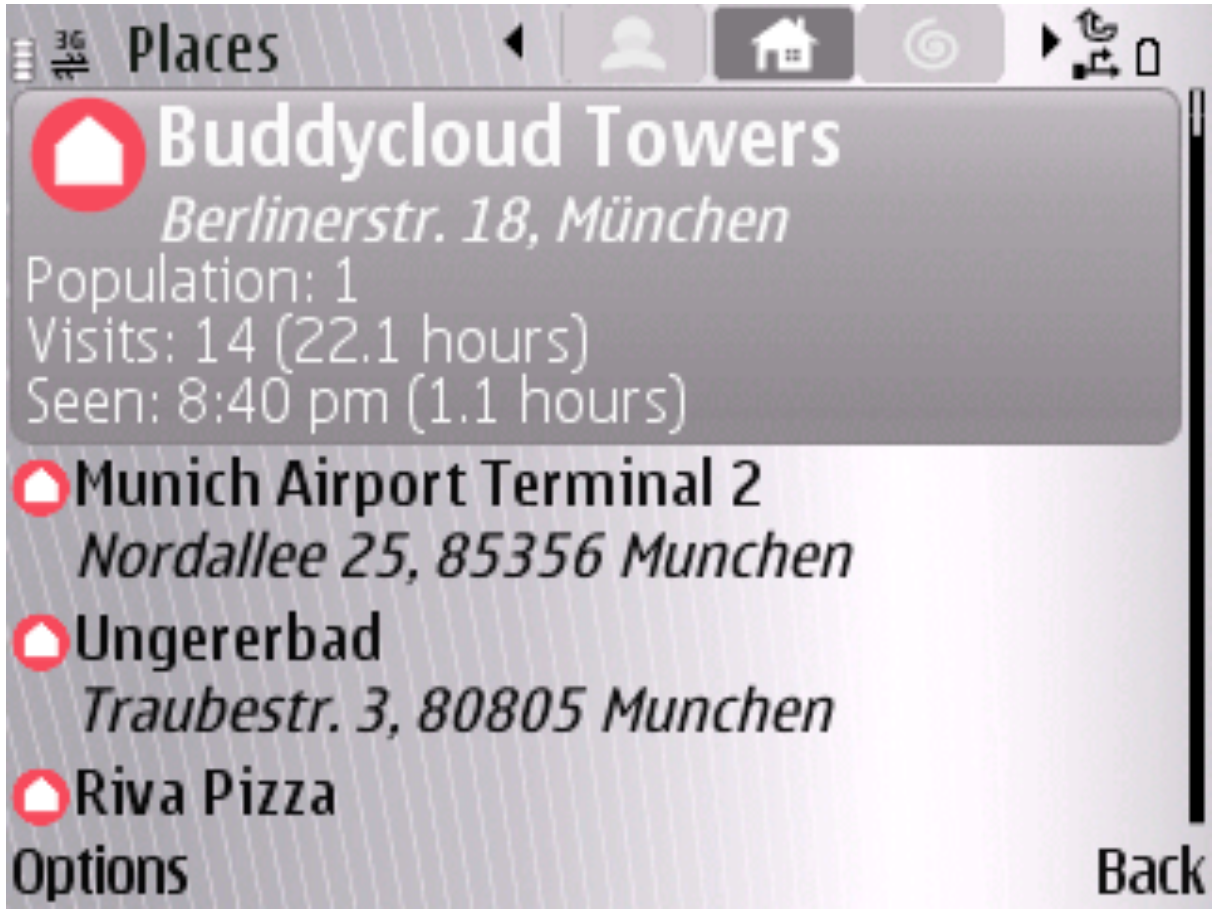
What can you do with XMPP?

Voice & Video



What can you do with XMPP?

Mobile apps / Geolocation



What can you do with XMPP?

<insert your idea here>

XMPP ...

XMPP ...

- ... is an open standard (RFC 3920 + 3921, XSF Extensions)

XMPP ...

- ... is an open standard (RFC 3920 + 3921, XSF Extensions)
- ... is decentralized (federated)

XMPP ...

- ... is an open standard (RFC 3920 + 3921, XSF Extensions)
- ... is decentralized (federated)
- ... has strong security (TLS)

XMPP ...

- ... is an open standard (RFC 3920 + 3921, XSF Extensions)
- ... is decentralized (federated)
- ... has strong security (TLS)
- ... has lots of open-source projects (but other licensing allowed)

XMPP ...

- ... is an open standard (RFC 3920 + 3921, XSF Extensions)
- ... is decentralized (federated)
- ... has strong security (TLS)
- ... has lots of open-source projects (but other licensing allowed)
- ... has an active, open community

What does XMPP provide?

What does XMPP provide?

- Channel encryption and authentication

What does XMPP provide?

- Channel encryption and authentication
- Presence and contact lists

What does XMPP provide?

- Channel encryption and authentication
- Presence and contact lists
- One-to-one and multi-party messaging

What does XMPP provide?

- Channel encryption and authentication
- Presence and contact lists
- One-to-one and multi-party messaging
- Alerts and notifications (PubSub)

What does XMPP provide?

- Channel encryption and authentication
- Presence and contact lists
- One-to-one and multi-party messaging
- Alerts and notifications (PubSub)
- Service discovery and device capabilities

What does XMPP provide?

- Channel encryption and authentication
- Presence and contact lists
- One-to-one and multi-party messaging
- Alerts and notifications (PubSub)
- Service discovery and device capabilities
- Peer-to-peer media sessions (Jingle)

What does XMPP provide?

- Channel encryption and authentication
- Presence and contact lists
- One-to-one and multi-party messaging
- Alerts and notifications (PubSub)
- Service discovery and device capabilities
- Peer-to-peer media sessions (Jingle)
- Data forms and remote commands

What does XMPP provide?

- Channel encryption and authentication
- Presence and contact lists
- One-to-one and multi-party messaging
- Alerts and notifications (PubSub)
- Service discovery and device capabilities
- Peer-to-peer media sessions (Jingle)
- Data forms and remote commands
- And more (lots of extensions)

History

History

- Invented by Jeremie Miller as Jabber (1998)

History

- Invented by Jeremie Miller as Jabber (1998)
- First server + clients + libraries (1999-2000)

History

- Invented by Jeremie Miller as Jabber (1998)
- First server + clients + libraries (1999-2000)
- More open source + commercial codebases (2000+)

History

- Invented by Jeremie Miller as Jabber (1998)
- First server + clients + libraries (1999-2000)
- More open source + commercial codebases (2000+)
- Core standardization in IETF as XMPP (2002-2004)

History

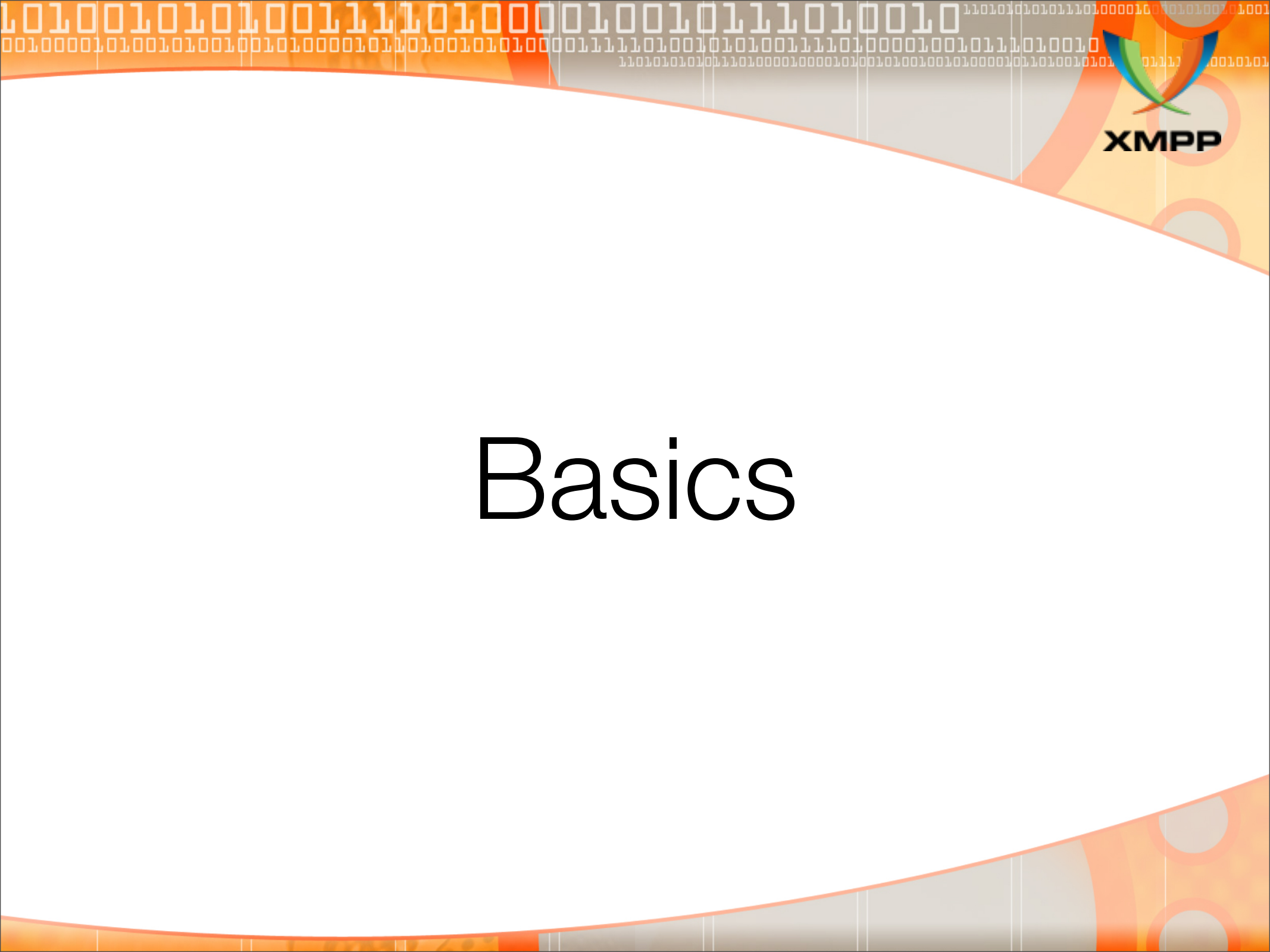
- Invented by Jeremie Miller as Jabber (1998)
- First server + clients + libraries (1999-2000)
- More open source + commercial codebases (2000+)
- Core standardization in IETF as XMPP (2002-2004)
- Development of extensions (2002+)

History

- Invented by Jeremie Miller as Jabber (1998)
- First server + clients + libraries (1999-2000)
- More open source + commercial codebases (2000+)
- Core standardization in IETF as XMPP (2002-2004)
- Development of extensions (2002+)
- Serious adoption by Apple, Google, LiveJournal, Nokia, Cisco, etc. (2005+)

History

- Invented by Jeremie Miller as Jabber (1998)
- First server + clients + libraries (1999-2000)
- More open source + commercial codebases (2000+)
- Core standardization in IETF as XMPP (2002-2004)
- Development of extensions (2002+)
- Serious adoption by Apple, Google, LiveJournal, Nokia, Cisco, etc. (2005+)
- Continuing work on improved security and more application types



Basics

Architecture

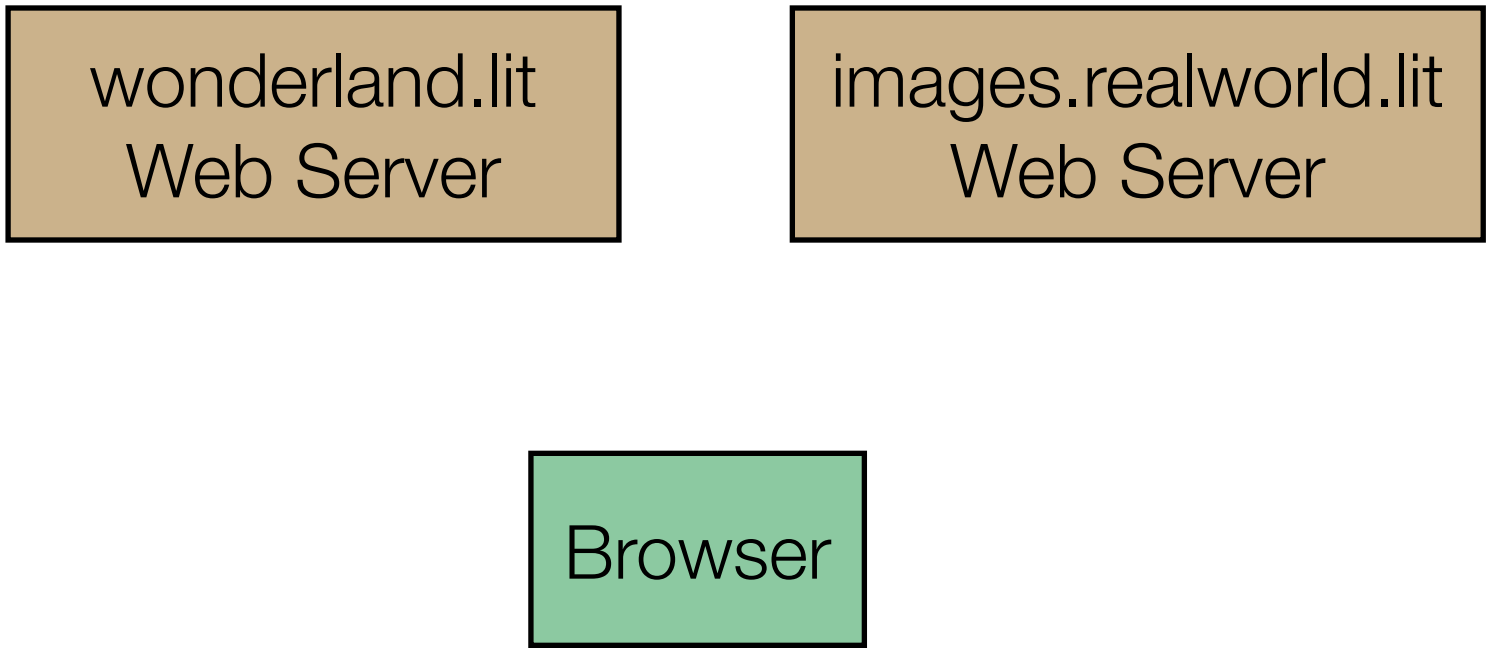
- Web architecture

Architecture

- Web architecture

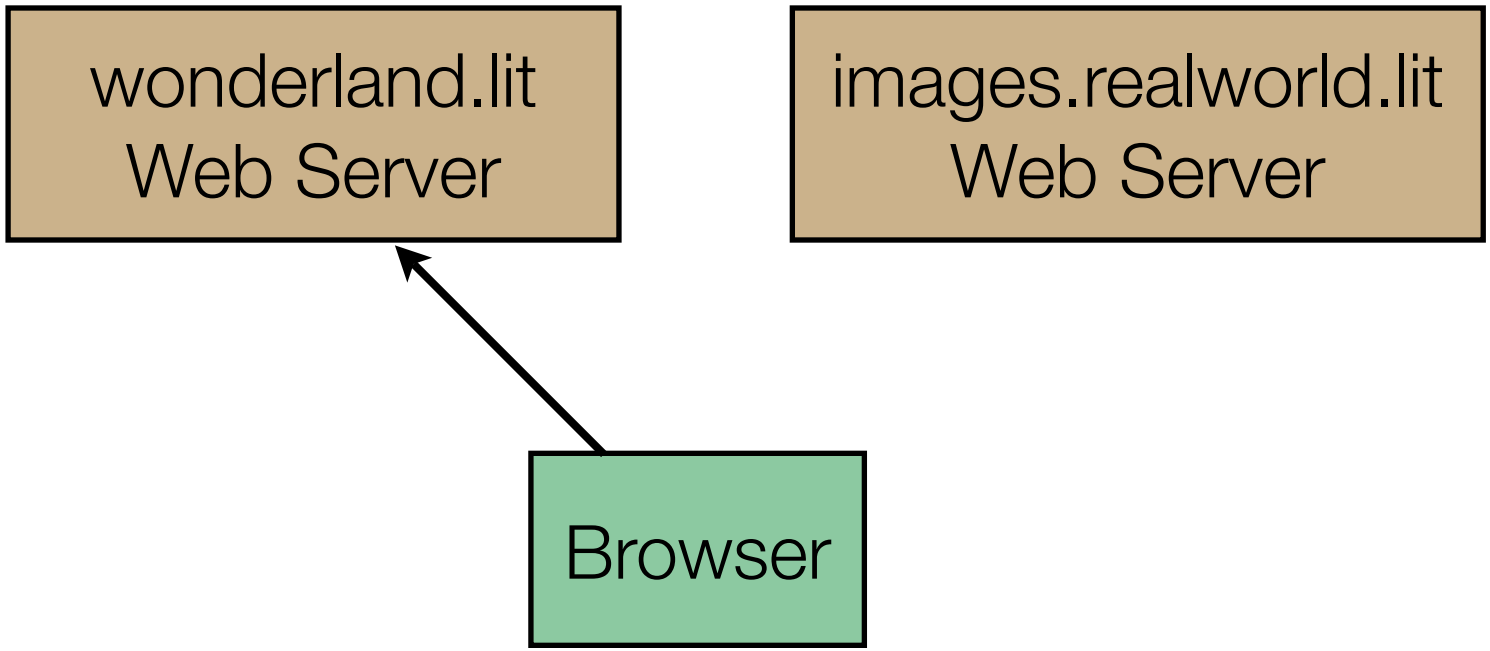
Architecture

- Web architecture



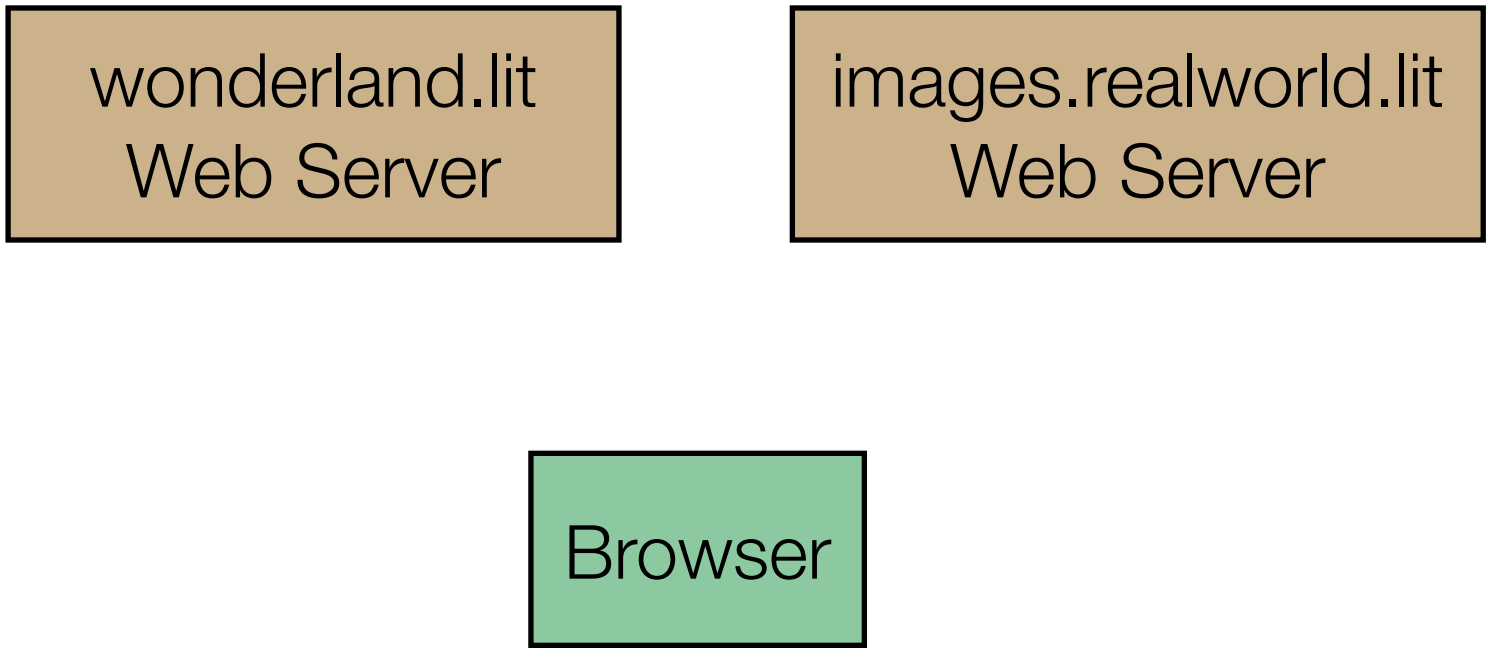
Architecture

- Web architecture



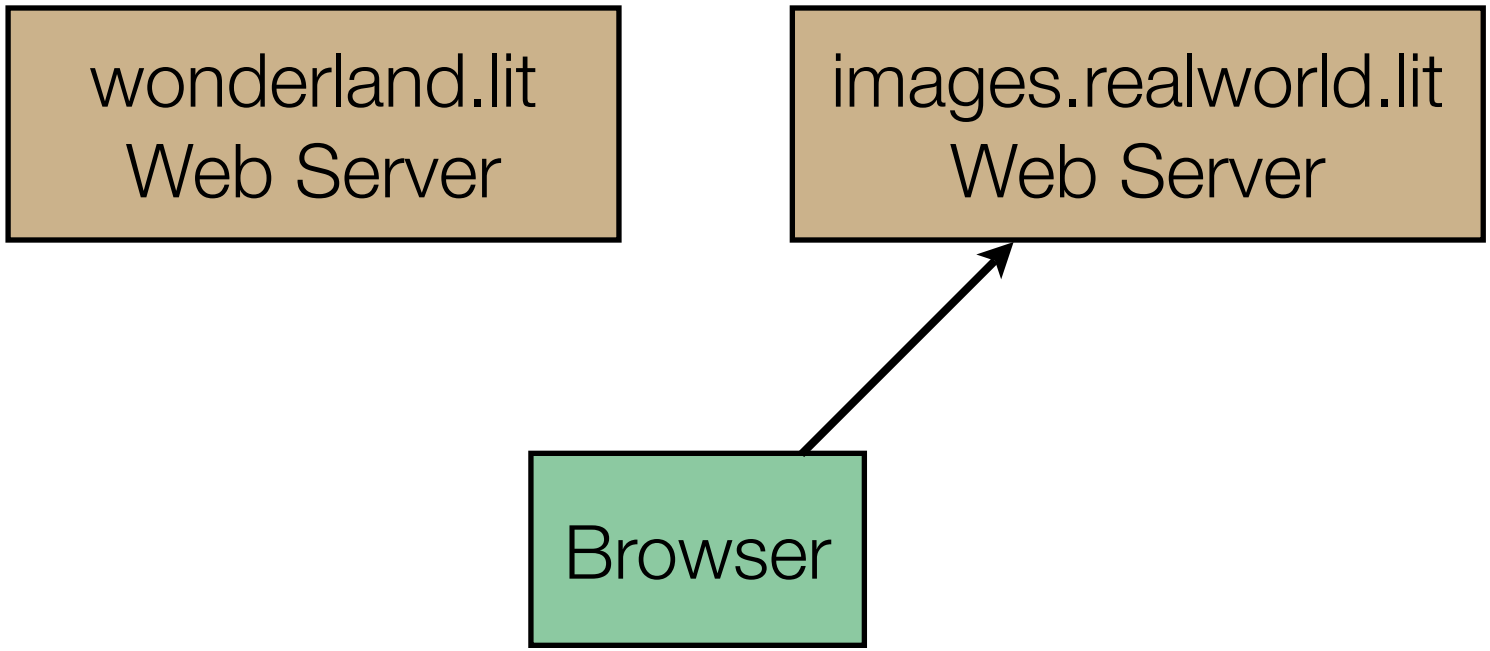
Architecture

- Web architecture



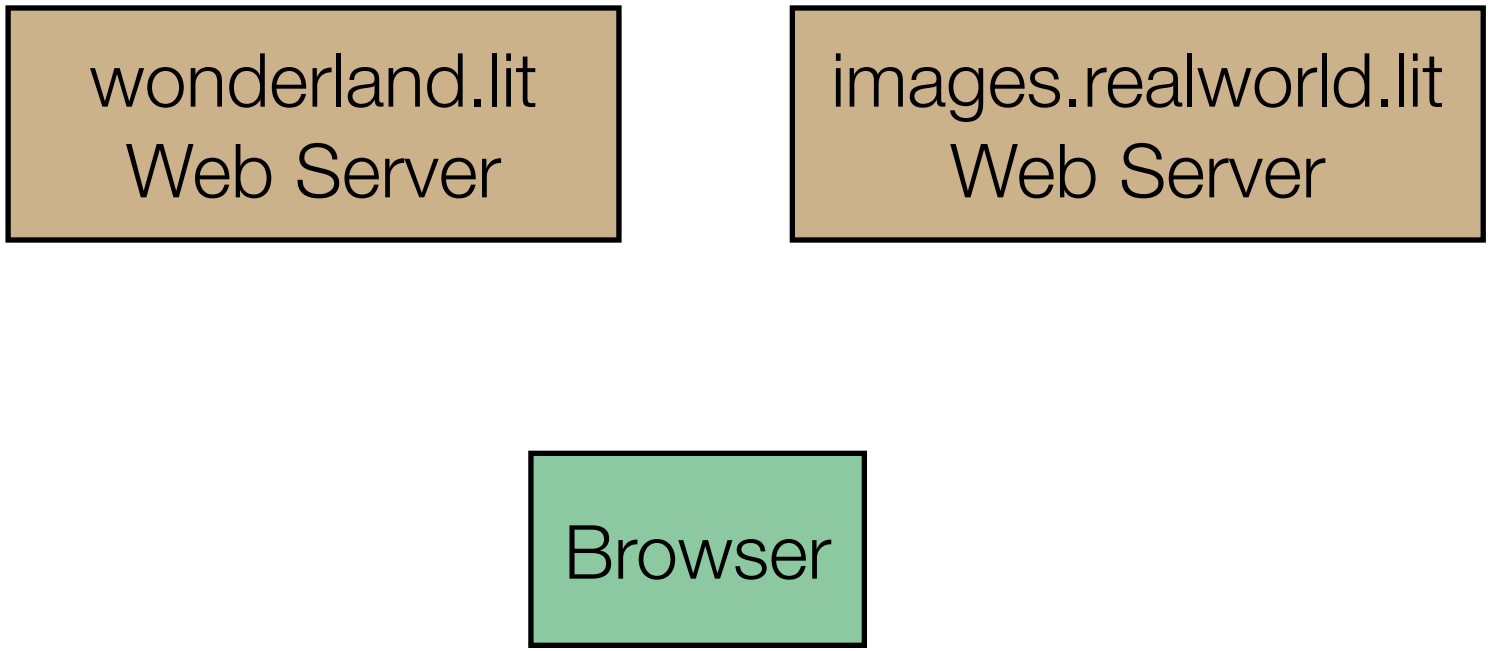
Architecture

- Web architecture



Architecture

- Web architecture



Architecture

- E-Mail architecture

Architecture

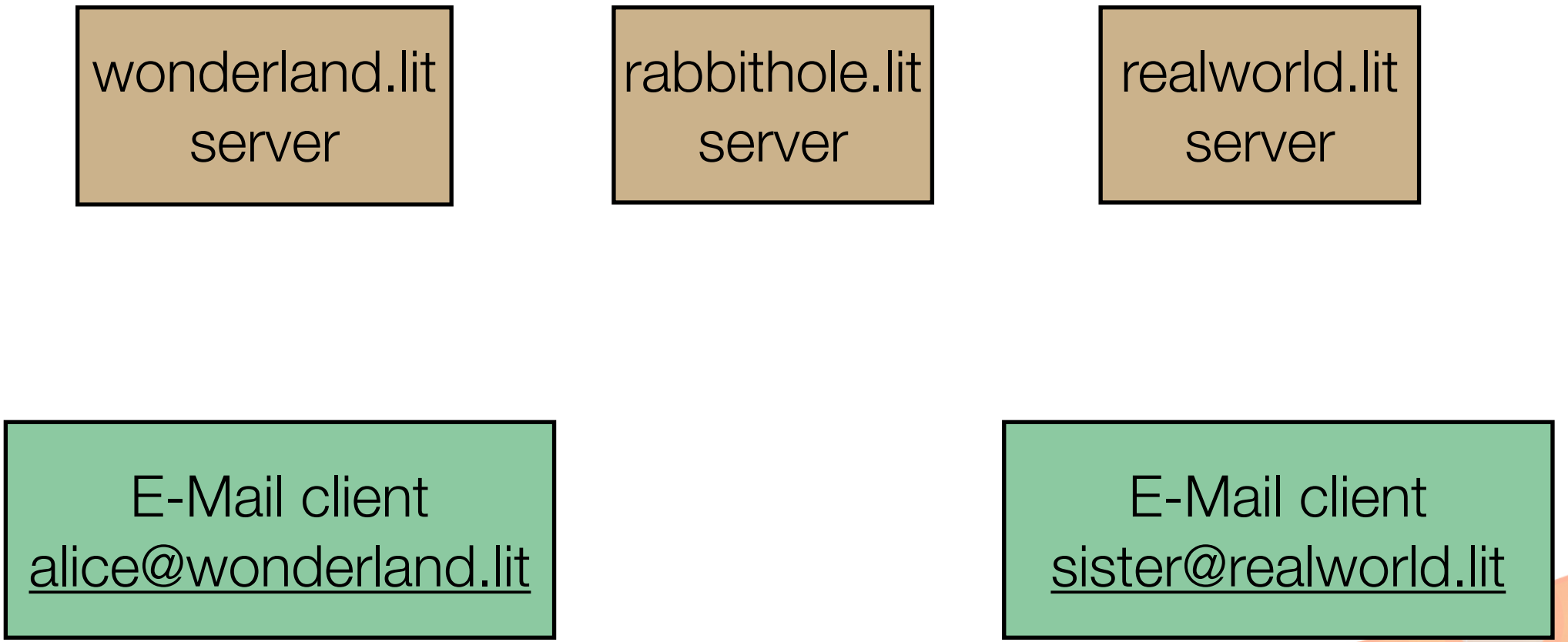
- E-Mail architecture

E-Mail client
alice@wonderland.lit

E-Mail client
sister@realworld.lit

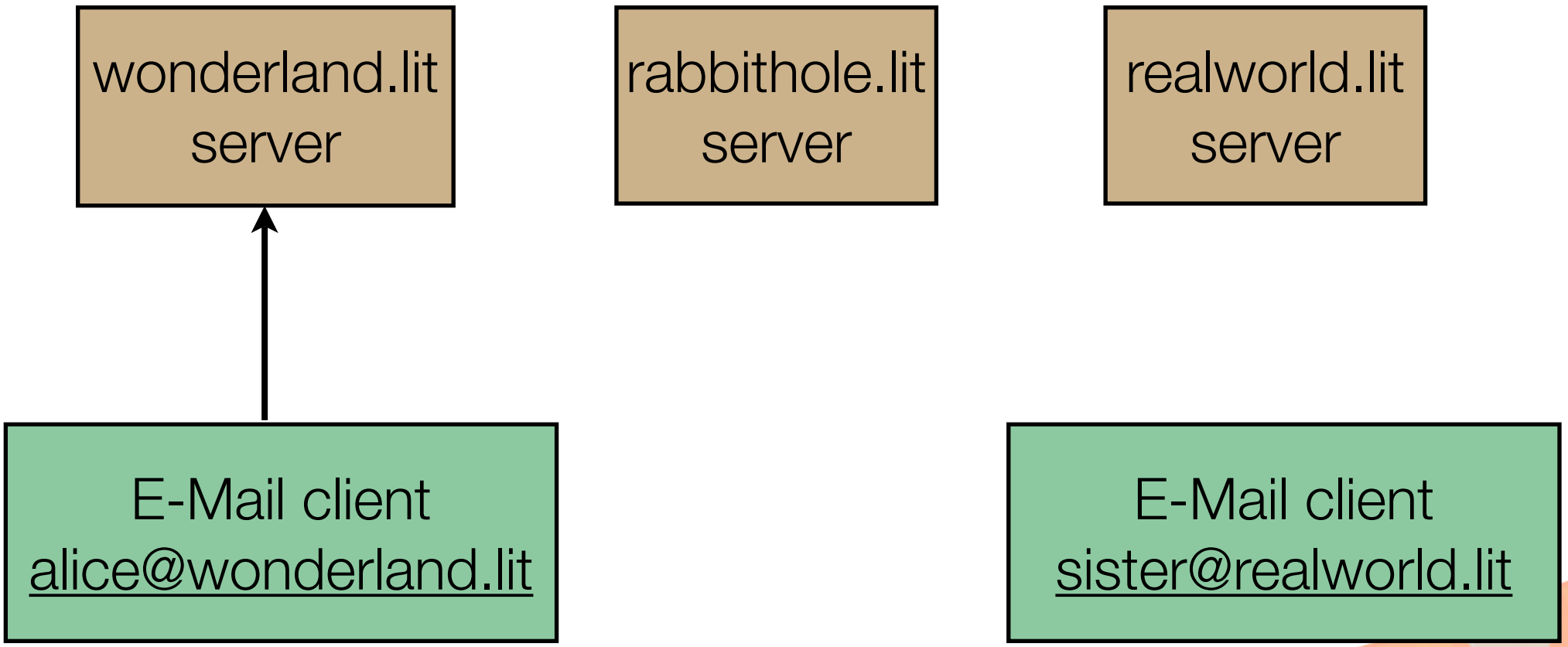
Architecture

- E-Mail architecture



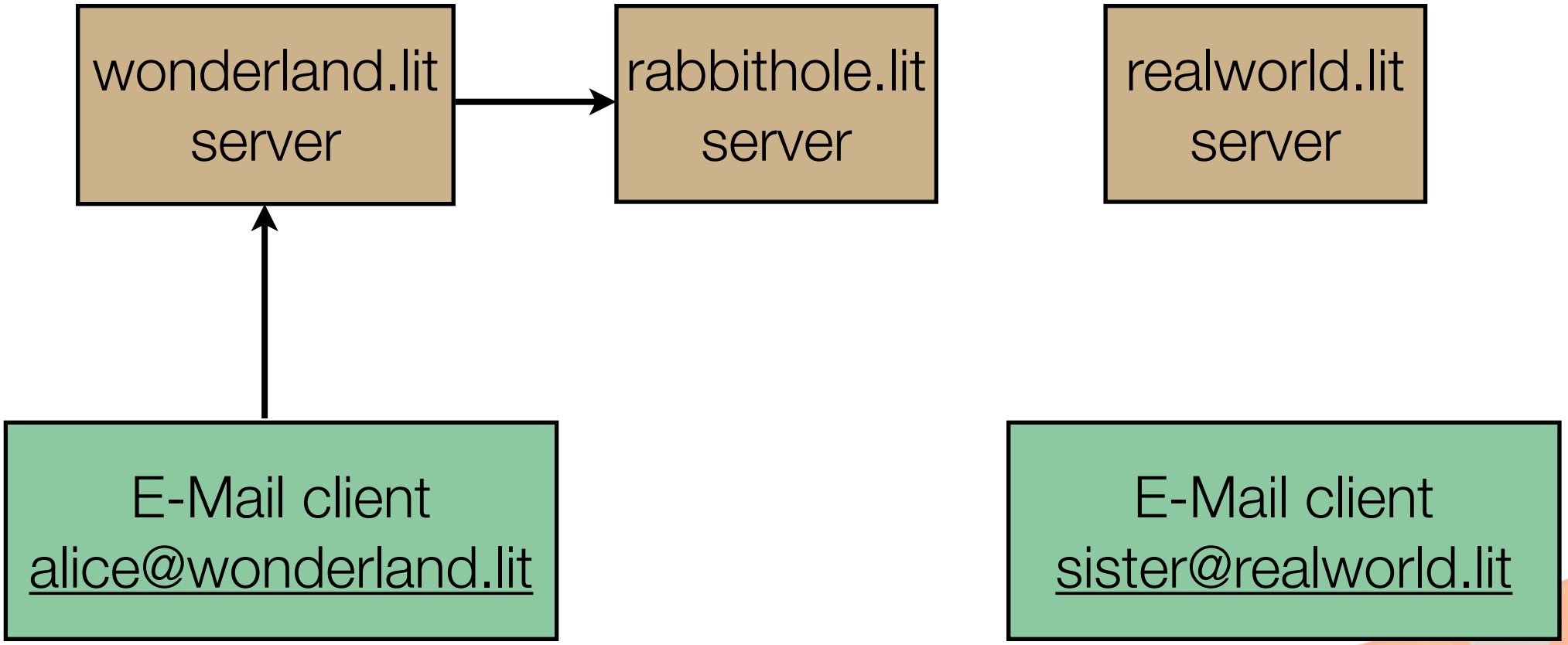
Architecture

- E-Mail architecture



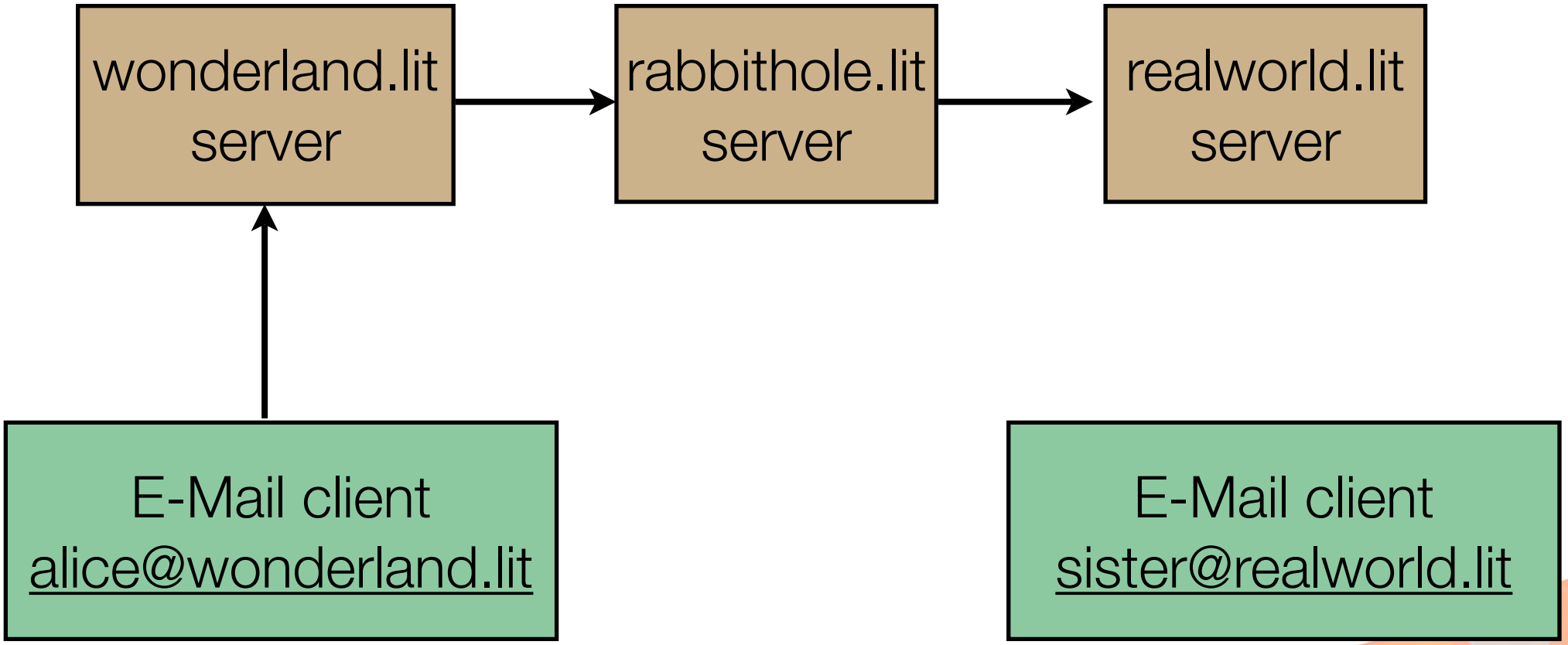
Architecture

- E-Mail architecture



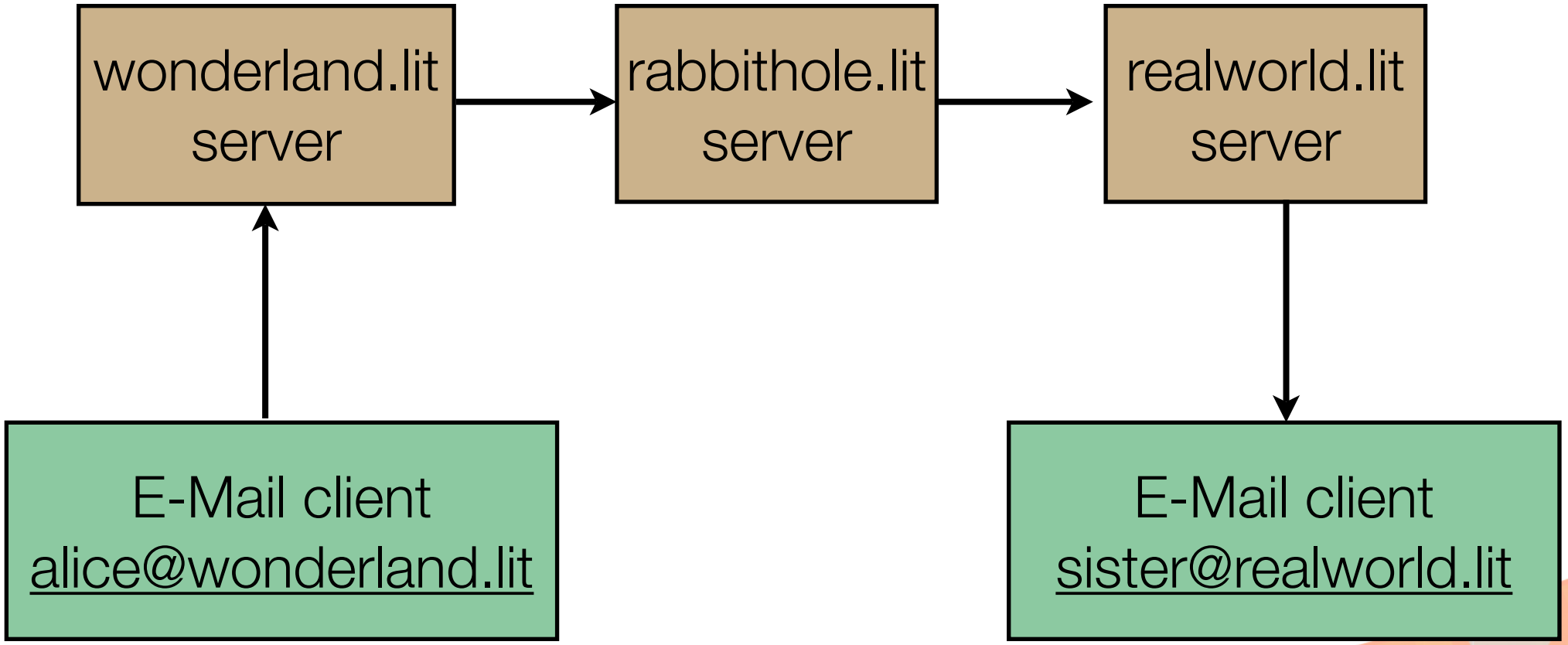
Architecture

- E-Mail architecture



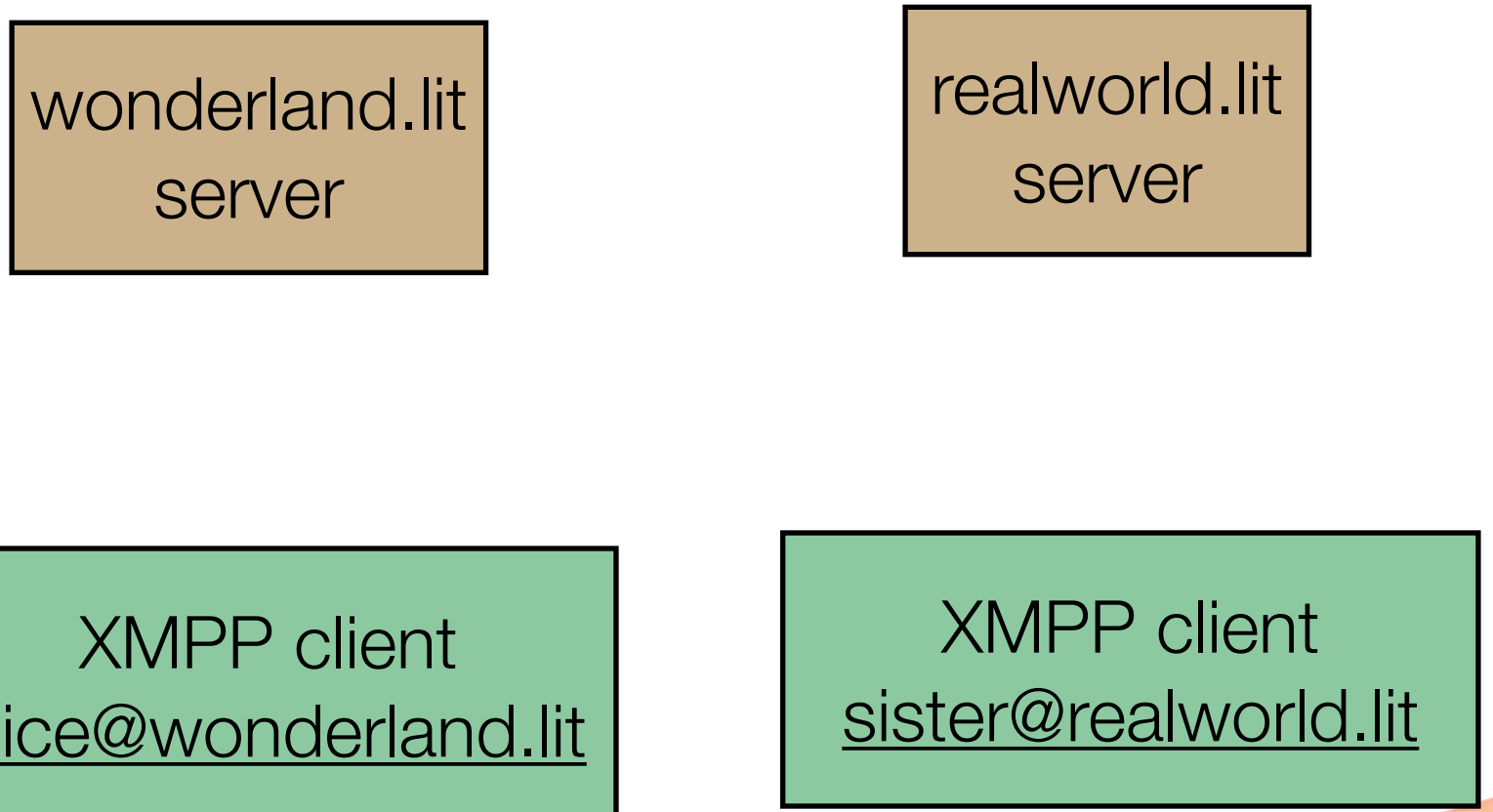
Architecture

- E-Mail architecture



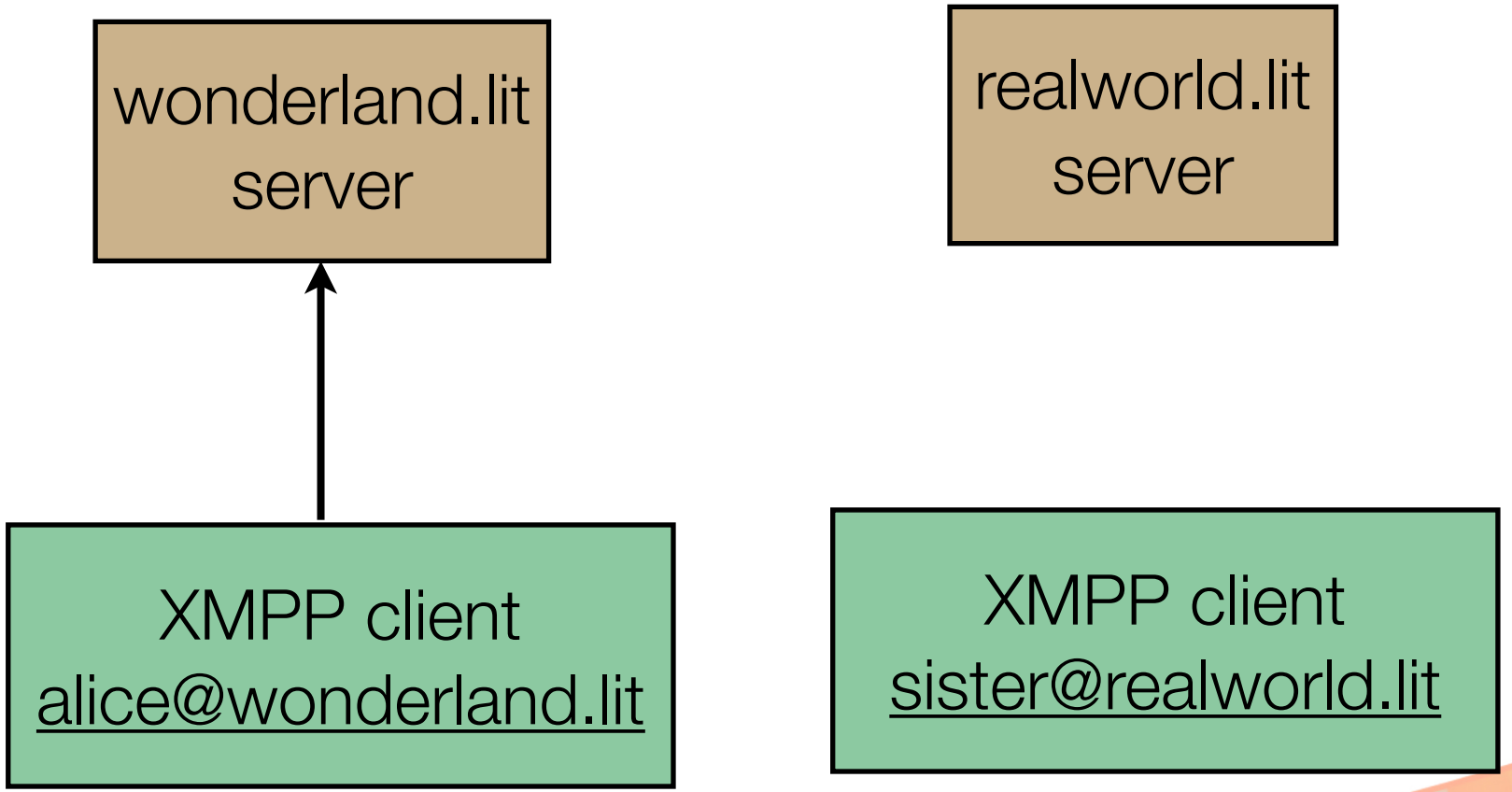
Architecture

- XMPP Architecture



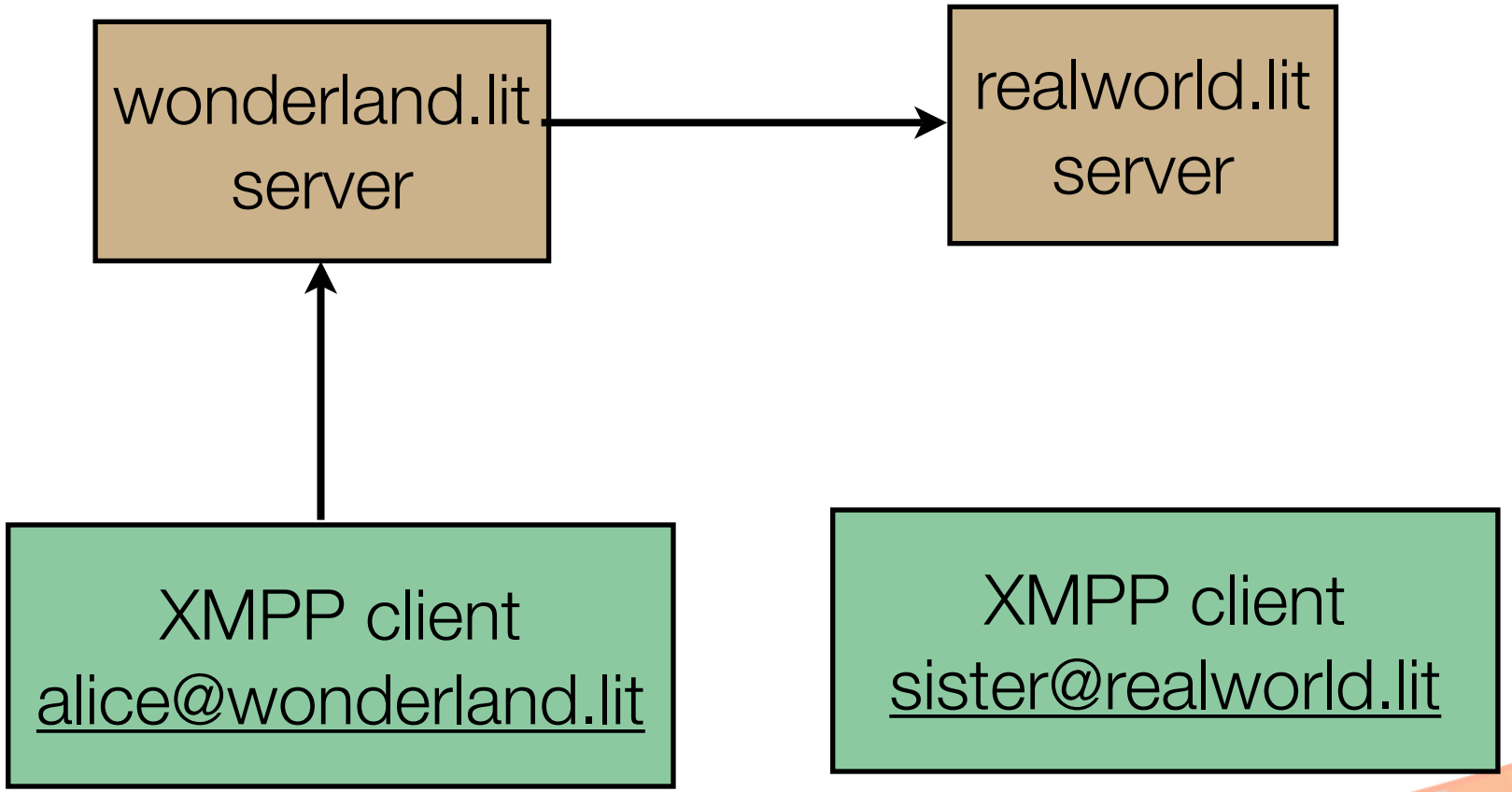
Architecture

- XMPP Architecture



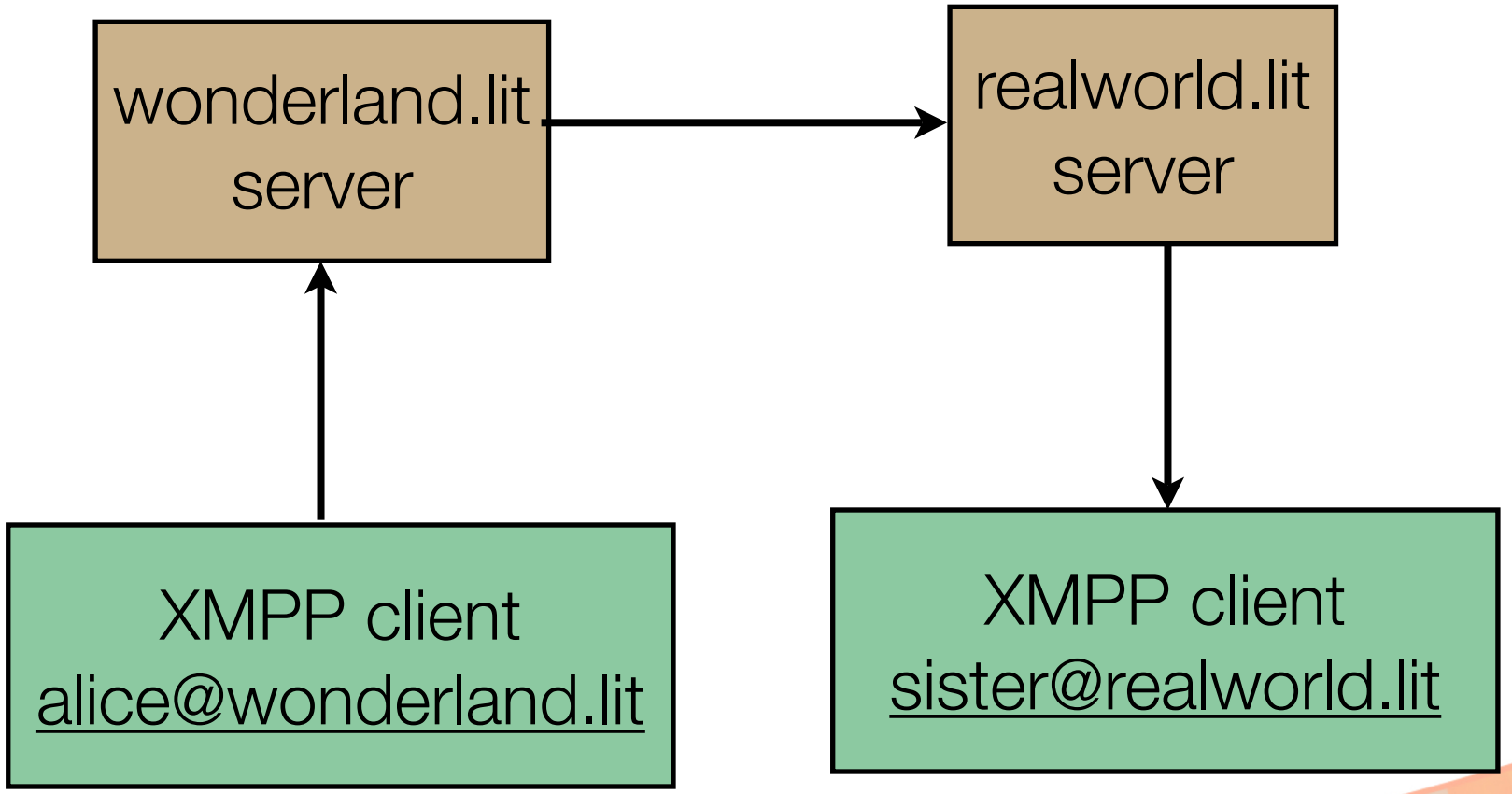
Architecture

- XMPP Architecture



Architecture

- XMPP Architecture



Architecture

Architecture

- Client-server

Architecture

- Client-server
  - Client developers can focus on user experience

Architecture

- Client-server
  - Client developers can focus on user experience
  - Server developers can focus on relaibility & scalability

Architecture

- Client-server
  - Client developers can focus on user experience
  - Server developers can focus on relaibility & scalability
- Decentralized

Architecture

- Client-server
  - Client developers can focus on user experience
  - Server developers can focus on relaibility & scalability
- Decentralized
  - Robust (no single point of failure)

Architecture

- Client-server
  - Client developers can focus on user experience
  - Server developers can focus on relaibility & scalability
- Decentralized
  - Robust (no single point of failure)
  - Easier to manage

Architecture

- Client-server
  - Client developers can focus on user experience
  - Server developers can focus on relaibility & scalability
- Decentralized
  - Robust (no single point of failure)
  - Easier to manage
- No multiple hops

Architecture

- Client-server
  - Client developers can focus on user experience
  - Server developers can focus on relaibility & scalability
- Decentralized
  - Robust (no single point of failure)
  - Easier to manage
- No multiple hops
  - Harder to spoof

Addresses

Jabber ID (JID)

Addresses

XMPP

Jabber ID (JID)

Domain

`wonderland.lit`

Addresses

Jabber ID (JID)

User Domain

alice@wonderland.lit

Addresses

Jabber ID (JID)

User Domain

alice@wonderland.lit

Bare JID

Addresses

Jabber ID (JID)

User Domain

alice@wonderland.lit

Addresses

Jabber ID (JID)

User Domain Resource

alice@wonderland.lit/TeaParty

Addresses

Jabber ID (JID)

User Domain Resource

alice@wonderland.lit/TeaParty

Full JID

Streaming XML

Streaming XML


```
<stream:stream>
```

Streaming XML

```
<stream:stream>
    <presence/>
```

Streaming XML


```
<stream:stream>
    <presence/>
    <iq type="get">
      <query xmlns="jabber:iq:roster"/>
    </iq>
```

Streaming XML


```
<stream:stream>
    <presence/>
    <iq type="get">
      <query xmlns="jabber:iq:roster"/>
    </iq>
    <iq type="result">
      <query xmlns="jabber:iq:roster">
        <item jid="alice@wonderland.lit"/>
        <item jid="madhatter@wonderland.lit"/>
        <item jid="whiterabbit@wonderland.lit"/>
      </query>
    </iq>
```

Streaming XML

```
<stream:stream>
    <presence/>
    <iq type="get">
      <query xmlns="jabber:iq:roster"/>
    </iq>
    <iq type="result">
      <query xmlns="jabber:iq:roster">
        <item jid="alice@wonderland.lit"/>
        <item jid="madhatter@wonderland.lit"/>
        <item jid="whiterabbit@wonderland.lit"/>
      </query>
    </iq>
    <message from="queen@wonderland.lit" to="madhatter@wonderland.lit">
      <body>Off with his head!</body>
    </message>
```

Streaming XML


```
<stream:stream>
    <presence/>
    <iq type="get">
      <query xmlns="jabber:iq:roster"/>
    </iq>
    <iq type="result">
      <query xmlns="jabber:iq:roster">
        <item jid="alice@wonderland.lit"/>
        <item jid="madhatter@wonderland.lit"/>
        <item jid="whiterabbit@wonderland.lit"/>
      </query>
    </iq>
    <message from="queen@wonderland.lit" to="madhatter@wonderland.lit">
      <body>Off with his head!</body>
    </message>
    <message from="king@wonderland.lit" to="party@rooms.wonderland.lit">
      <body>You are all pardoned.</body>
    </message>
```

Streaming XML

XMPP

```
<stream:stream>
  <presence/>
  <iq type="get">
    <query xmlns="jabber:iq:roster"/>
  </iq>
  <iq type="result">
    <query xmlns="jabber:iq:roster">
      <item jid="alice@wonderland.lit"/>
      <item jid="madhatter@wonderland.lit"/>
      <item jid="whiterabbit@wonderland.lit"/>
    </query>
  </iq>
  <message from="queen@wonderland.lit" to="madhatter@wonderland.lit">
    <body>Off with his head!</body>
  </message>
  <message from="king@wonderland.lit" to="party@rooms.wonderland.lit">
    <body>You are all pardoned.</body>
  </message>
  <presence type="unavailable"/>
```

Streaming XML

```
<stream:stream>
    <presence/>
    <iq type="get">
      <query xmlns="jabber:iq:roster"/>
    </iq>
    <iq type="result">
      <query xmlns="jabber:iq:roster">
        <item jid="alice@wonderland.lit"/>
        <item jid="madhatter@wonderland.lit"/>
        <item jid="whiterabbit@wonderland.lit"/>
      </query>
    </iq>
    <message from="queen@wonderland.lit" to="madhatter@wonderland.lit">
      <body>Off with his head!</body>
    </message>
    <message from="king@wonderland.lit" to="party@rooms.wonderland.lit">
      <body>You are all pardoned.</body>
    </message>
    <presence type="unavailable"/>
</stream:stream>
```

Communication Primitives

Stanzas:

Communication Primitives

Stanzas:

<message/>

Communication Primitives

Stanzas:

<message/>

<presence/>

Communication Primitives

Stanzas:

<message/>

<presence/>

<iq/>

Message Stanzas

Message Stanzas

```
<message from="madhatter@wonderland.lit/foo"
         to="alice@wonderland.lit"
         type="chat">
  <body>Who are you?</body>
  <subject>Query</subject>
</message>
```

Message Stanzas

```
<message from="madhatter@wonderland.lit/foo"
         to="alice@wonderland.lit"
         type="chat">
  <body>Who are you?</body>
  <subject>Query</subject>
</message>
```

- From, To

Message Stanzas

```
<message from="madhatter@wonderland.lit/foo"
         to="alice@wonderland.lit"
         type="chat">
  <body>Who are you?</body>
  <subject>Query</subject>
</message>
```

- From, To
- Types: normal, chat, groupchat, headline, error

Message Stanzas

```
<message from="madhatter@wonderland.lit/foo"
         to="alice@wonderland.lit"
         type="chat">
  <body>Who are you?</body>
  <subject>Query</subject>
</message>
```

- From, To
- Types: normal, chat, groupchat, headline, error
- Payloads: Body, Subject

Presence Stanzas

Presence Stanzas

```
<presence from="alice@wonderland.lit/pda">
  <show>xa</show>
  <status>down the rabbit hole!</status>
</presence>
```

Presence Stanzas

```
<presence from="alice@wonderland.lit/pda">
  <show>xa</show>
  <status>down the rabbit hole!</status>
</presence>
```

- Advertise network availability (online/offline)

Presence Stanzas

```
<presence from="alice@wonderland.lit/pda">
  <show>xa</show>
  <status>down the rabbit hole!</status>
</presence>
```

- Advertise network availability (online/offline)
- ‘Show’ Statuses: Away, Do Not Disturb, Extended Away, Free for chat

Presence Stanzas

```
<presence from="alice@wonderland.lit/pda">
  <show>xa</show>
  <status>down the rabbit hole!</status>
</presence>
```

- Advertise network availability (online/offline)
- ‘Show’ Statuses: Away, Do Not Disturb, Extended Away, Free for chat
- Status messages

Presence Stanzas

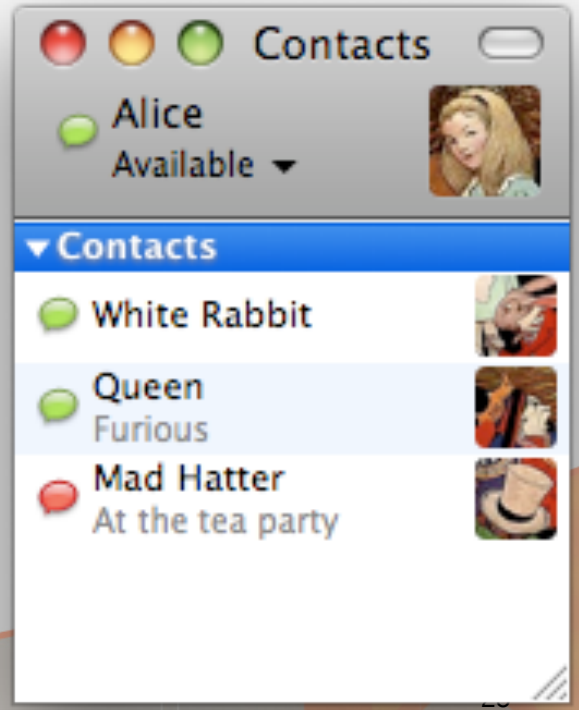
```
<presence from="alice@wonderland.lit/pda">
  <show>xa</show>
  <status>down the rabbit hole!</status>
</presence>
```

- Advertise network availability (online/offline)
- ‘Show’ Statuses: Away, Do Not Disturb, Extended Away, Free for chat
- Status messages
- Typically used for contact lists/rosters

Presence Stanzas

```
<presence from="alice@wonderland.lit/pda">
  <show>xa</show>
  <status>down the rabbit hole!</status>
</presence>
```

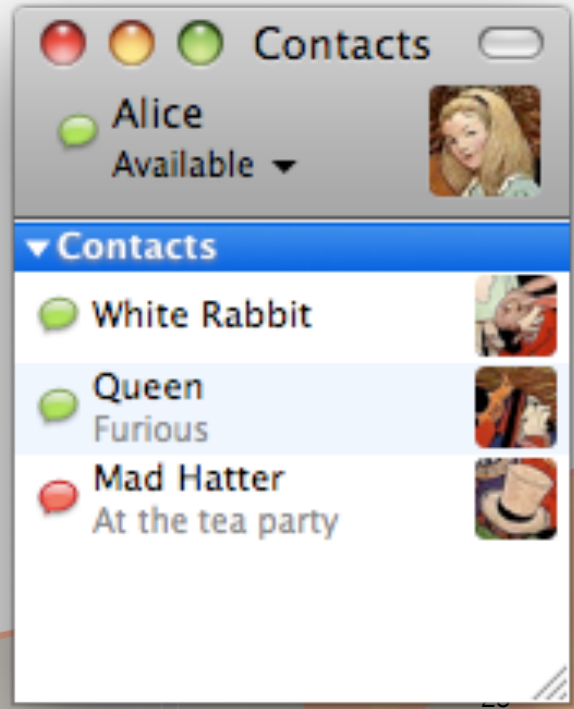
- Advertise network availability (online/offline)
- ‘Show’ Statuses: Away, Do Not Disturb, Extended Away, Free for chat
- Status messages
- Typically used for contact lists/rosters



Presence Stanzas

```
<presence from="alice@wonderland.lit/pda">
  <show>xa</show>
  <status>down the rabbit hole!</status>
</presence>
```

- Advertise network availability (online/offline)
- ‘Show’ Statuses: Away, Do Not Disturb, Extended Away, Free for chat
- Status messages
- Typically used for contact lists/rosters
- Presence subscriptions



IQ Stanzas

IQ Stanzas

```
<iq type="get">
  <query xmlns="jabber:iq:roster"/>
</iq>
```

IQ Stanzas

```
<iq type="get">
  <query xmlns="jabber:iq:roster"/>
</iq>

<iq type="result">
  <query xmlns="jabber:iq:roster">
    <item jid="alice@wonderland.lit"/>
    <item jid="madhatter@wonderland.lit"/>
    <item jid="whiterabbit@wonderland.lit"/>
  </query>
</iq>
```

IQ Stanzas

```
<iq type="get">
  <query xmlns="jabber:iq:roster"/>
</iq>

<iq type="result">
  <query xmlns="jabber:iq:roster">
    <item jid="alice@wonderland.lit"/>
    <item jid="madhatter@wonderland.lit"/>
    <item jid="whiterabbit@wonderland.lit"/>
  </query>
</iq>
```

- Request/Response

IQ Stanzas

```
<iq type="get">
  <query xmlns="jabber:iq:roster"/>
</iq>

<iq type="result">
  <query xmlns="jabber:iq:roster">
    <item jid="alice@wonderland.lit"/>
    <item jid="madhatter@wonderland.lit"/>
    <item jid="whiterabbit@wonderland.lit"/>
  </query>
</iq>
```

- Request/Response
- Workflows, execute commands, query information

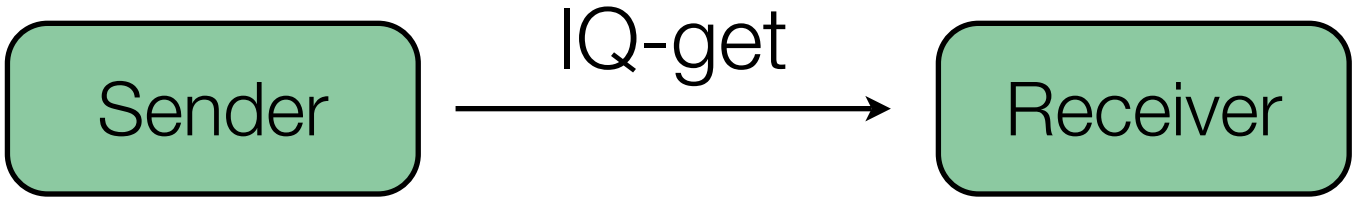
IQ Stanzas

```
<iq type="get">
  <query xmlns="jabber:iq:roster"/>
</iq>

<iq type="result">
  <query xmlns="jabber:iq:roster">
    <item jid="alice@wonderland.lit"/>
    <item jid="madhatter@wonderland.lit"/>
    <item jid="whiterabbit@wonderland.lit"/>
  </query>
</iq>
```

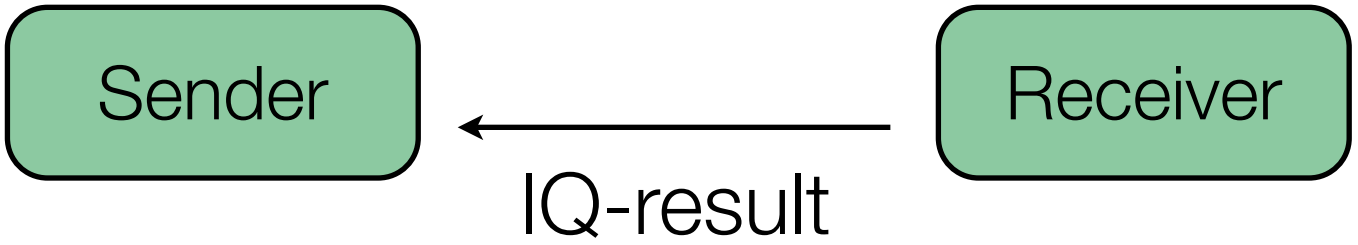
- Request/Response
- Workflows, execute commands, query information
- Similar to HTTP GET, POST, PUT

IQ Stanzas

IQ Stanzas

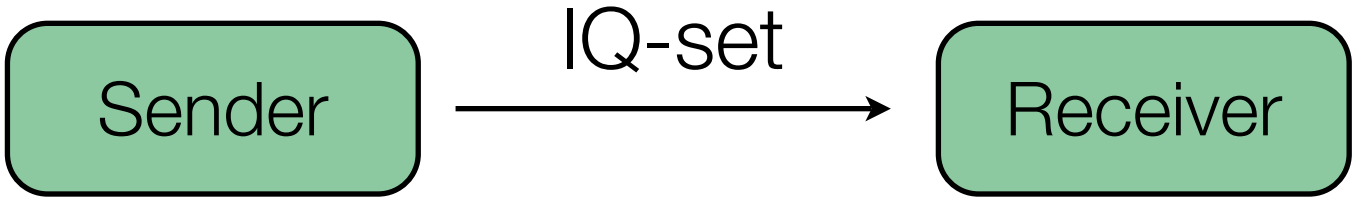


IQ Stanzas

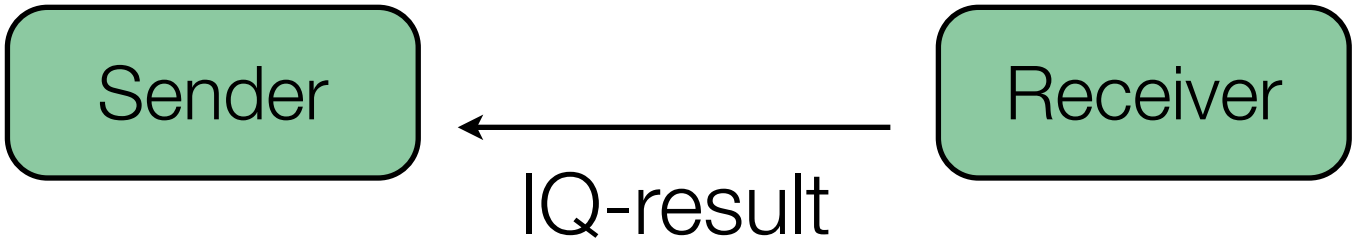


IQ Stanzas

IQ Stanzas



IQ Stanzas



IQ Stanzas

IQ Stanzas

IQ Stanzas

```
<iq type="get">
  <query xmlns="jabber:iq:roster"/>
</iq>
```

IQ Stanzas

```
<iq type="get">
  <query xmlns="jabber:iq:roster"/>
</iq>
```

- Types:

IQ Stanzas

```
<iq type="get">
  <query xmlns="jabber:iq:roster"/>
</iq>
```

- Types:
  - get: Ask for information (HTTP GET)

IQ Stanzas

```
<iq type="get">
  <query xmlns="jabber:iq:roster"/>
</iq>
```

- Types:
  - get: Ask for information (HTTP GET)
  - set: Provide information (HTTP POST/PUT)

IQ Stanzas

```
<iq type="get">
  <query xmlns="jabber:iq:roster"/>
</iq>
```

- Types:
  - get: Ask for information (HTTP GET)
  - set: Provide information (HTTP POST/PUT)
  - result: Returns requested information / acknowledge set

IQ Stanzas

```
<iq type="get">
  <query xmlns="jabber:iq:roster"/>
</iq>
```

- Types:
  - get: Ask for information (HTTP GET)
  - set: Provide information (HTTP POST/PUT)
  - result: Returns requested information / acknowledge set
  - error

Extensibility

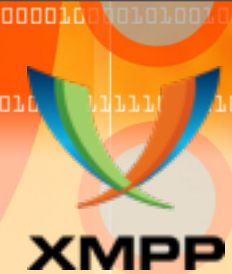
Extensibility

- Any XML child element can be used as a payload

Extensibility

- Any XML child element can be used as a payload
  - e.g. XHTML bodies, Atom feeds, XML-RPC, ...

Extensibility



- Any XML child element can be used as a payload
  - e.g. XHTML bodies, Atom feeds, XML-RPC, ...
- Namespaces to scope payloads

```
<iq type="get">
  <query xmlns="jabber:iq:roster"/>
</iq>
```

Extensibility

- Any XML child element can be used as a payload
  - e.g. XHTML bodies, Atom feeds, XML-RPC, ...
- Namespaces to scope payloads

```
<iq type="get">
  <query xmlns="jabber:iq:roster"/>
</iq>
```

- Extensions typically developed @ XMPP Standards Foundation's

Extensibility

- Any XML child element can be used as a payload
  - e.g. XHTML bodies, Atom feeds, XML-RPC, ...
- Namespaces to scope payloads

```
<iq type="get">
  <query xmlns="jabber:iq:roster"/>
</iq>
```

- Extensions typically developed @ XMPP Standards Foundation's
- Open, developer-friendly standards process

Extensibility

- Any XML child element can be used as a payload
  - e.g. XHTML bodies, Atom feeds, XML-RPC, ...
- Namespaces to scope payloads

```
<iq type="get">
  <query xmlns="jabber:iq:roster"/>
</iq>
```

- Extensions typically developed @ XMPP Standards Foundation's
- Open, developer-friendly standards process
- Can write your own "private" extensions for custom functionality

Asynchronicity

Asynchronicity

- Web

Asynchronicity

- Web
  - Send request to server

Asynchronicity

- Web
  - Send request to server
  - Wait for response

Asynchronicity

- Web
  - Send request to server
  - Wait for response
  - Short-lived connections

Asynchronicity

- Web
  - Send request to server
  - Wait for response
  - Short-lived connections
- Jabber

Asynchronicity

- Web
  - Send request to server
  - Wait for response
  - Short-lived connections
- Jabber
  - Long lived connection

Asynchronicity

- Web
  - Send request to server
  - Wait for response
  - Short-lived connections
- Jabber
  - Long lived connection
  - Events are sent out / come in asynchronously

Asynchronicity

- Web
  - Send request to server
  - Wait for response
  - Short-lived connections
- Jabber
  - Long lived connection
  - Events are sent out / come in asynchronously
- Different mindsets!

Code Example

Echo bot

Echo bot

- Bots

Echo bot

- Bots
  - Unmanned clients

Echo bot

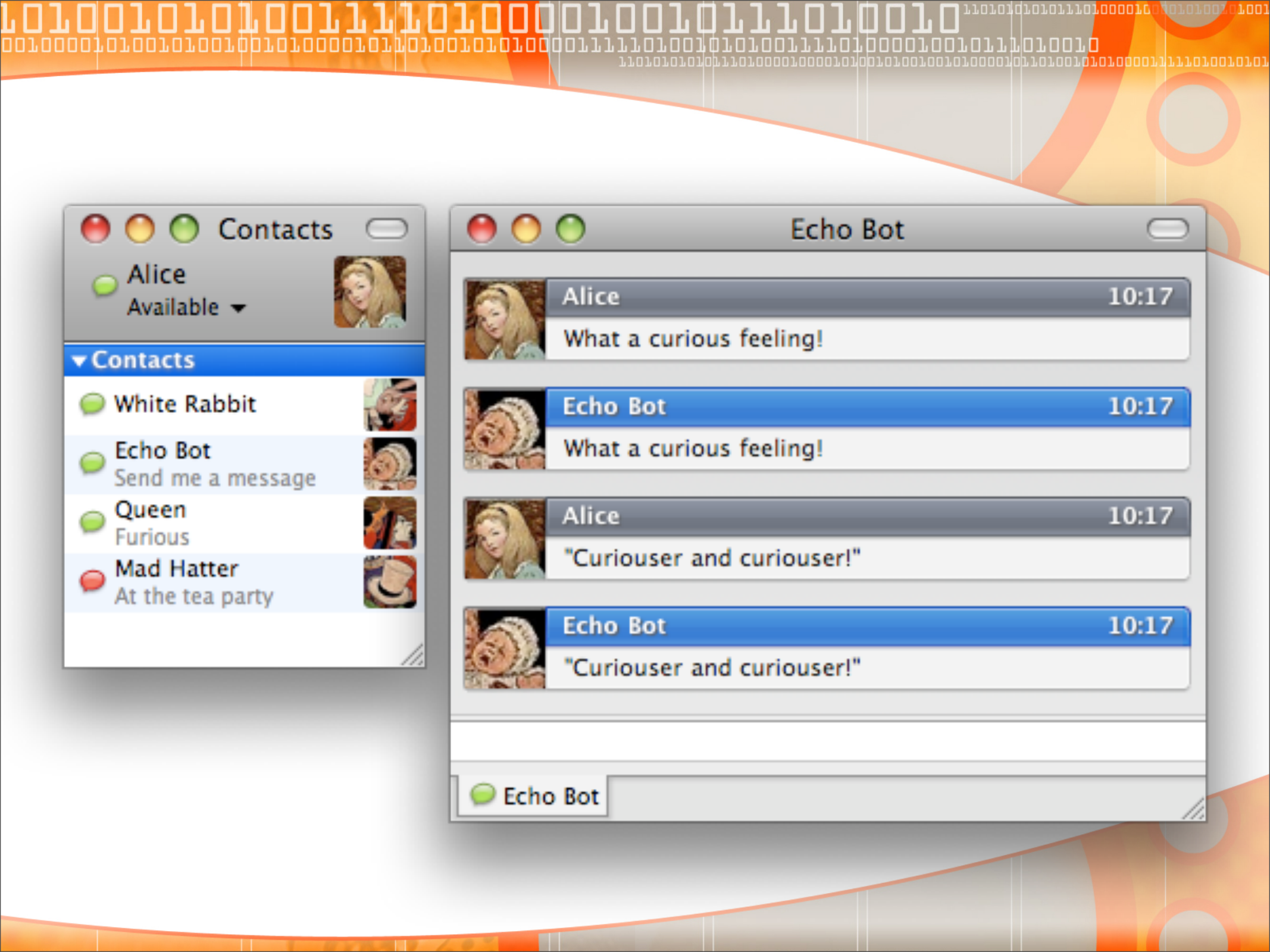
- Bots
  - Unmanned clients
  - Connect to an XMPP server, and wait for commands / send events

Echo bot

- Bots
  - Unmanned clients
  - Connect to an XMPP server, and wait for commands / send events
- Echo bot

Echo bot

- Bots
  - Unmanned clients
  - Connect to an XMPP server, and wait for commands / send events
- Echo bot
  - Echoes back every message that it receives


Contacts
Alice
Available
Contacts
White Rabbit
Echo Bot
Send me a message
Queen
Furious
Mad Hatter
At the tea party
Echo Bot
Alice 10:17
What a curious feeling!
Echo Bot 10:17
What a curious feeling!
Alice 10:17
"Curiouser and curiouser!"
Echo Bot 10:17
"Curiouser and curiouser!"
Echo Bot

Writing the Echo Bot

Writing the Echo Bot

- Select a language to work in

Writing the Echo Bot

- Select a language to work in
  - e.g. Python

Writing the Echo Bot

- Select a language to work in
  - e.g. Python
- Select an XMPP library to do the low-level XMPP work

Writing the Echo Bot

- Select a language to work in
  - e.g. Python
- Select an XMPP library to do the low-level XMPP work
  - e.g. SleekXMPP

Writing the Echo Bot

- Select a language to work in
  - e.g. Python
- Select an XMPP library to do the low-level XMPP work
  - e.g. SleekXMPP
- Start coding

SleekXMPP Python Echo Bot


```
class EchoBot :
  def __init__(self, jid, password) :
    self.xmpp = sleekxmpp.ClientXMPP(jid, password)
    self.xmpp.add_event_handler("session_start", self.handleXMPPConnected)
    self.xmpp.add_event_handler("message", self.handleIncomingMessage)

  def run(self) :
    self.xmpp.connect()
    self.xmpp.process(threaded=False)

  def handleXMPPConnected(self, event):
    self.xmpp.sendPresence(pstatus = "Send me a message")

  def handleIncomingMessage(self, message) :
    self.xmpp.sendMessage(message["jid"], message["message"])

def main() :
  bot = EchoBot("echobot@wonderland.lit/HelloWorld", "mypass")
  bot.run()
```

SleekXMPP Python Echo Bot


```
class EchoBot :
  def __init__(self, jid, password) :
    self.xmpp = sleekxmpp.ClientXMPP(jid, password)
    self.xmpp.add_event_handler("session_start", self.handleXMPPConnected)
    self.xmpp.add_event_handler("message", self.handleIncomingMessage)

  def run(self) :
    self.xmpp.connect()
    self.xmpp.process(threaded=False)

  def handleXMPPConnected(self, event):
    self.xmpp.sendPresence(pstatus = "Send me a message")

  def handleIncomingMessage(self, message) :
    self.xmpp.sendMessage(message["jid"], message["message"])

def main() :
  bot = EchoBot("echobot@wonderland.lit/HelloWorld", "mypass")
  bot.run()
```

SleekXMPP Python Echo Bot


```
class EchoBot :
  def __init__(self, jid, password) :
    self.xmpp = sleekxmpp.ClientXMPP(jid, password)
    self.xmpp.add_event_handler("session_start", self.handleXMPPConnected)
    self.xmpp.add_event_handler("message", self.handleIncomingMessage)

  def run(self) :
    self.xmpp.connect()
    self.xmpp.process(threaded=False)

  def handleXMPPConnected(self, event):
    self.xmpp.sendPresence(pstatus = "Send me a message")

  def handleIncomingMessage(self, message) :
    self.xmpp.sendMessage(message["jid"], message["message"])

def main() :
  bot = EchoBot("echobot@wonderland.lit/HelloWorld", "mypass")
  bot.run()
```

SleekXMPP Python Echo Bot


```
class EchoBot :
  def __init__(self, jid, password) :
    self.xmpp = sleekxmpp.ClientXMPP(jid, password)
    self.xmpp.add_event_handler("session_start", self.handleXMPPConnected)
    self.xmpp.add_event_handler("message", self.handleIncomingMessage)

  def run(self) :
    self.xmpp.connect()
    self.xmpp.process(threaded=False)

  def handleXMPPConnected(self, event):
    self.xmpp.sendPresence(pstatus = "Send me a message")

  def handleIncomingMessage(self, message) :
    self.xmpp.sendMessage(message["jid"], message["message"])

def main() :
  bot = EchoBot("echobot@wonderland.lit/HelloWorld", "mypass")
  bot.run()
```

SleekXMPP Python Echo Bot

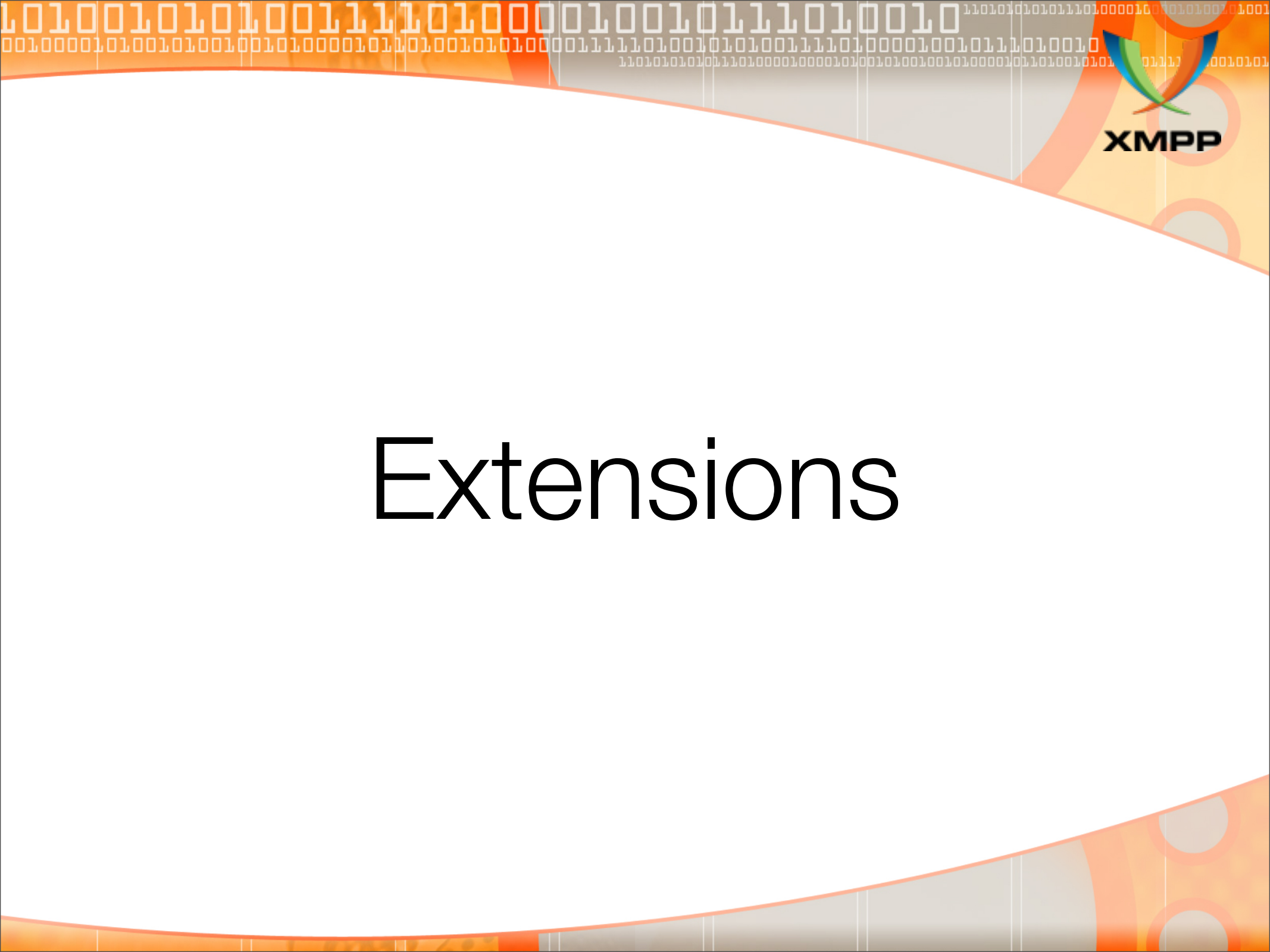

```
class EchoBot :
  def __init__(self, jid, password) :
    self.xmpp = sleekxmpp.ClientXMPP(jid, password)
    self.xmpp.add_event_handler("session_start", self.handleXMPPConnected)
    self.xmpp.add_event_handler("message", self.handleIncomingMessage)

  def run(self) :
    self.xmpp.connect()
    self.xmpp.process(threaded=False)

  def handleXMPPConnected(self, event):
    self.xmpp.sendPresence(pstatus = "Send me a message")

  def handleIncomingMessage(self, message) :
    self.xmpp.sendMessage(message["jid"], message["message"])

def main() :
  bot = EchoBot("echobot@wonderland.lit/HelloWorld", "mypass")
  bot.run()
```

SleekXMPP Python Echo Bot


```
class EchoBot :
  def __init__(self, jid, password) :
    self.xmpp = sleekxmpp.ClientXMPP(jid, password)
    self.xmpp.add_event_handler("session_start", self.handleXMPPConnected)
    self.xmpp.add_event_handler("message", self.handleIncomingMessage)

  def run(self) :
    self.xmpp.connect()
    self.xmpp.process(threaded=False)

  def handleXMPPConnected(self, event):
    self.xmpp.sendPresence(pstatus = "Send me a message")

  def handleIncomingMessage(self, message) :
    self.xmpp.sendMessage(message["jid"], message["message"])

def main() :
  bot = EchoBot("echobot@wonderland.lit/HelloWorld", "mypass")
  bot.run()
```

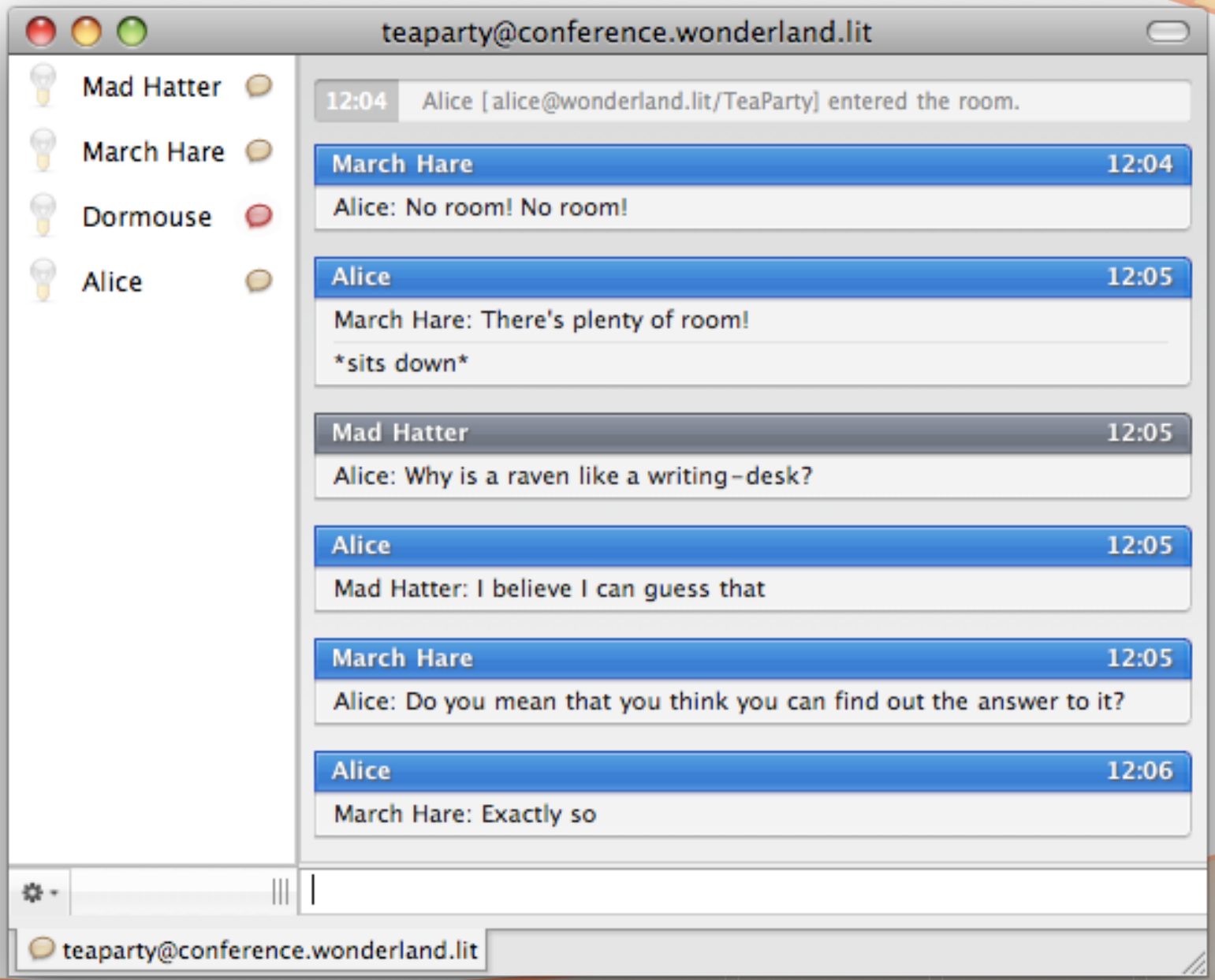
SleekXMPP Python Echo Bot


```
class EchoBot :
  def __init__(self, jid, password) :
    self.xmpp = sleekxmpp.ClientXMPP(jid, password)
    self.xmpp.add_event_handler("session_start", self.handleXMPPConnected)
    self.xmpp.add_event_handler("message", self.handleIncomingMessage)

  def run(self) :
    self.xmpp.connect()
    self.xmpp.process(threaded=False)

  def handleXMPPConnected(self, event):
    self.xmpp.sendPresence(pstatus = "Send me a message")

  def handleIncomingMessage(self, message) :
    self.xmpp.sendMessage(message["jid"], message["message"])

def main() :
  bot = EchoBot("echobot@wonderland.lit/HelloWorld", "mypass")
  bot.run()
```



Extensions

Multi-User Chat



Multi-User Chat

Multi-User Chat

- Conversation with multiple users

Multi-User Chat

- Conversation with multiple users
- Shared roster (with presence from all participating users)

Multi-User Chat

- Conversation with multiple users
- Shared roster (with presence from all participating users)
- Crowd control

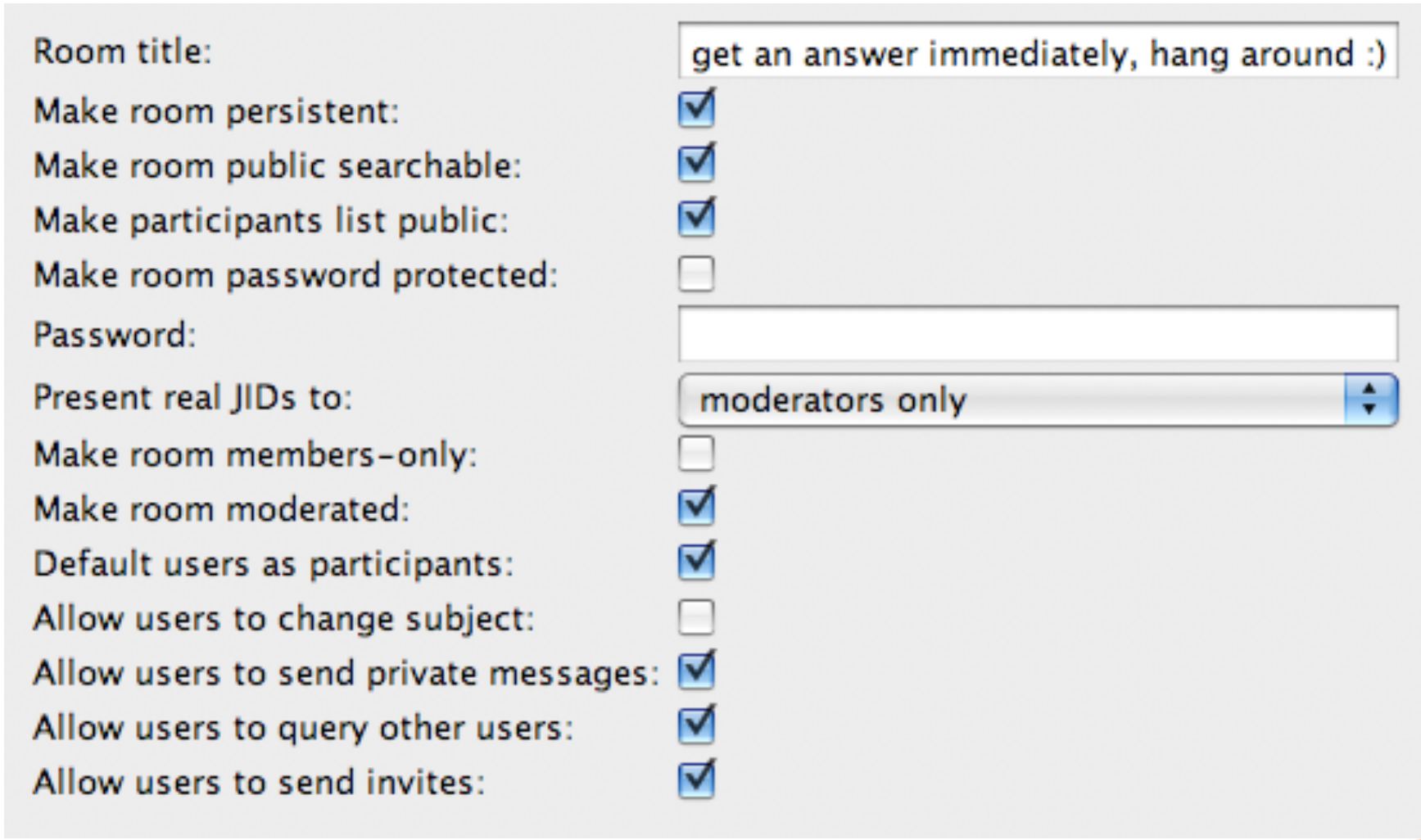
Multi-User Chat

- Conversation with multiple users
- Shared roster (with presence from all participating users)
- Crowd control
- Privacy

Multi-User Chat

- Conversation with multiple users
- Shared roster (with presence from all participating users)
- Crowd control
- Privacy
- Configurable

Multi-User Chat Configuration



PubSub: Polling vs PubSub

Polling

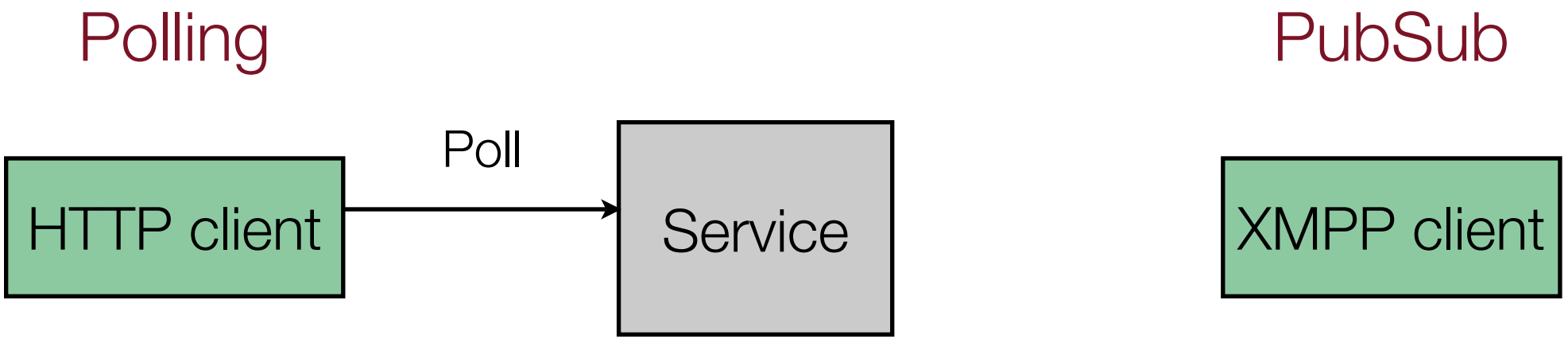
PubSub

HTTP client

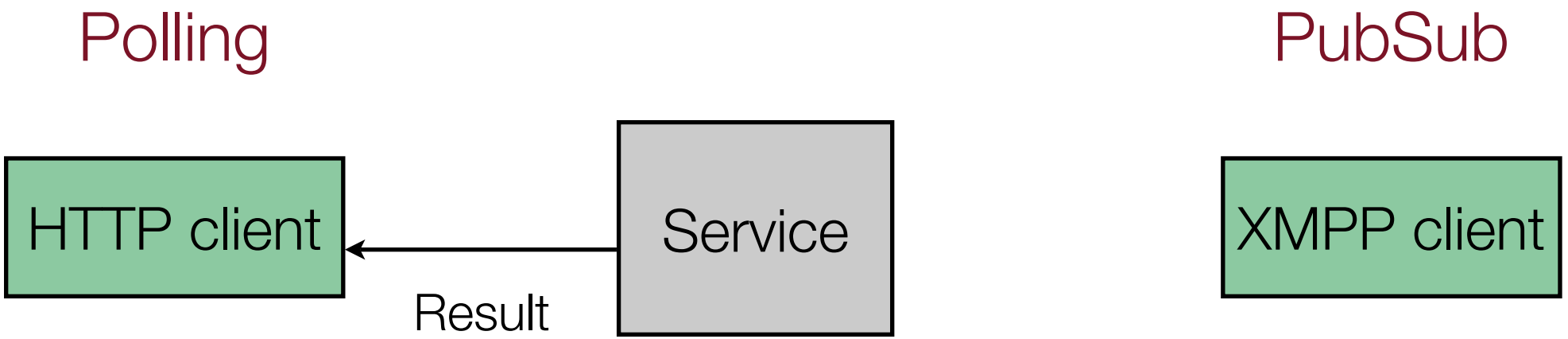
Service

XMPP client

PubSub: Polling vs PubSub



PubSub: Polling vs PubSub



PubSub: Polling vs PubSub

Polling

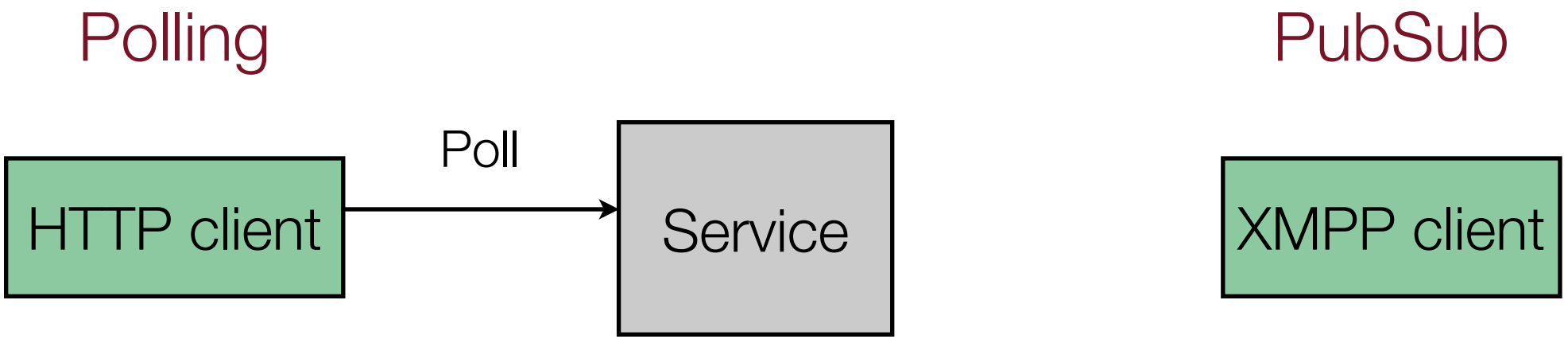
PubSub

HTTP client

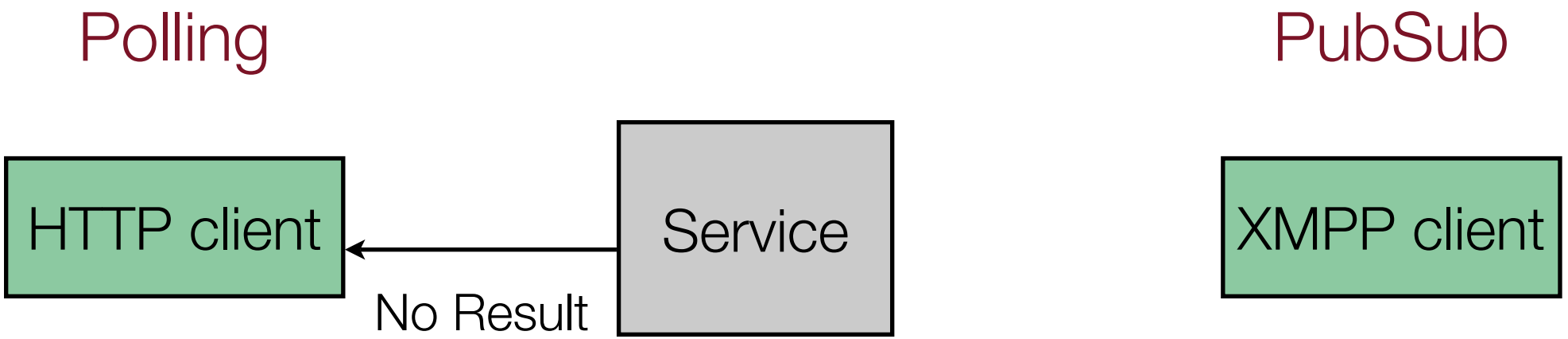
Service

XMPP client

PubSub: Polling vs PubSub



PubSub: Polling vs PubSub



PubSub: Polling vs PubSub

Polling

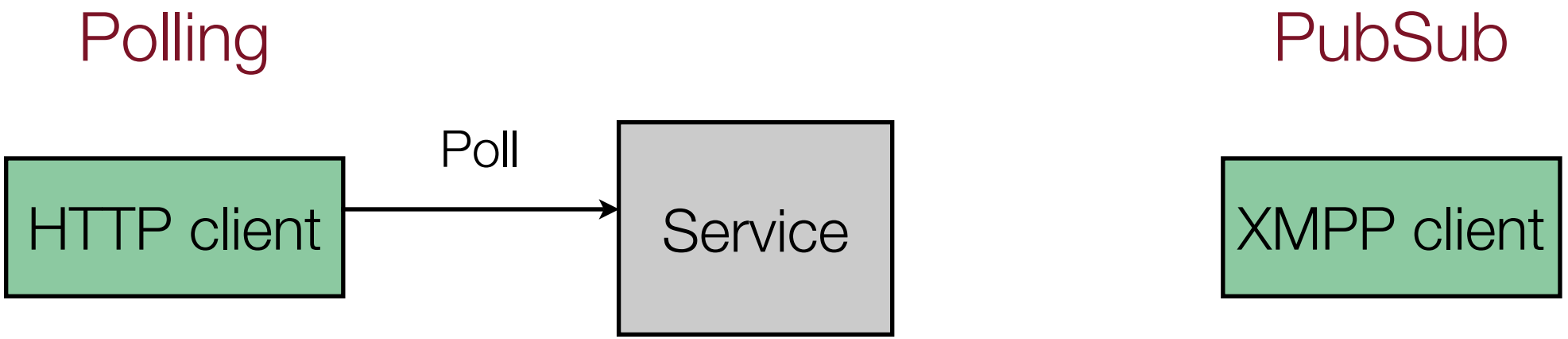
PubSub

HTTP client

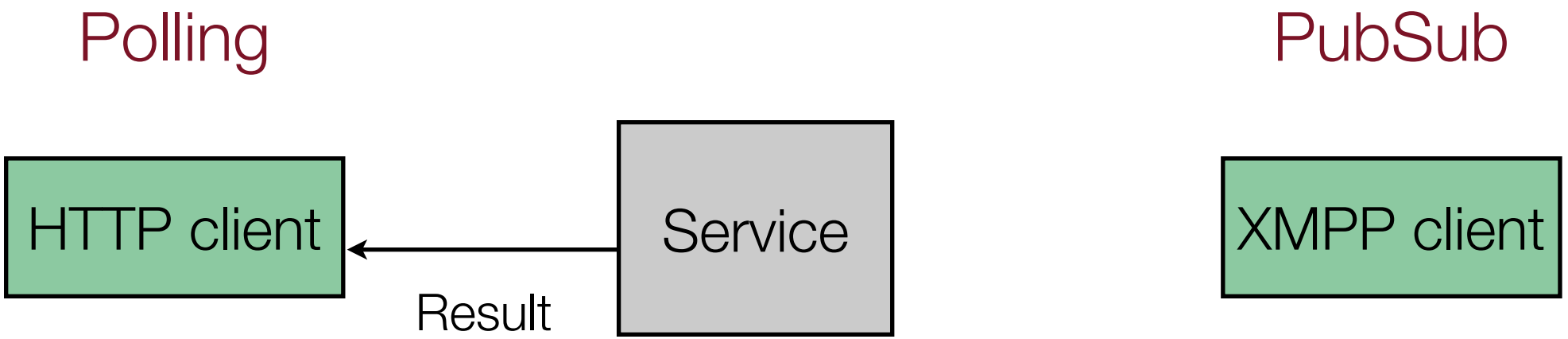
Service

XMPP client

PubSub: Polling vs PubSub



PubSub: Polling vs PubSub



PubSub: Polling vs PubSub

Polling

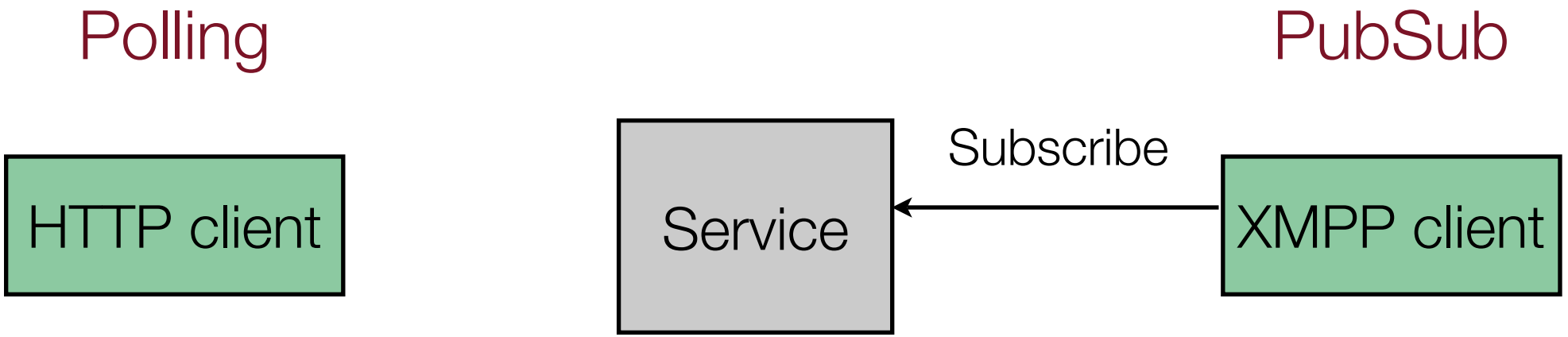
PubSub

HTTP client

Service

XMPP client

PubSub: Polling vs PubSub



PubSub: Polling vs PubSub

Polling

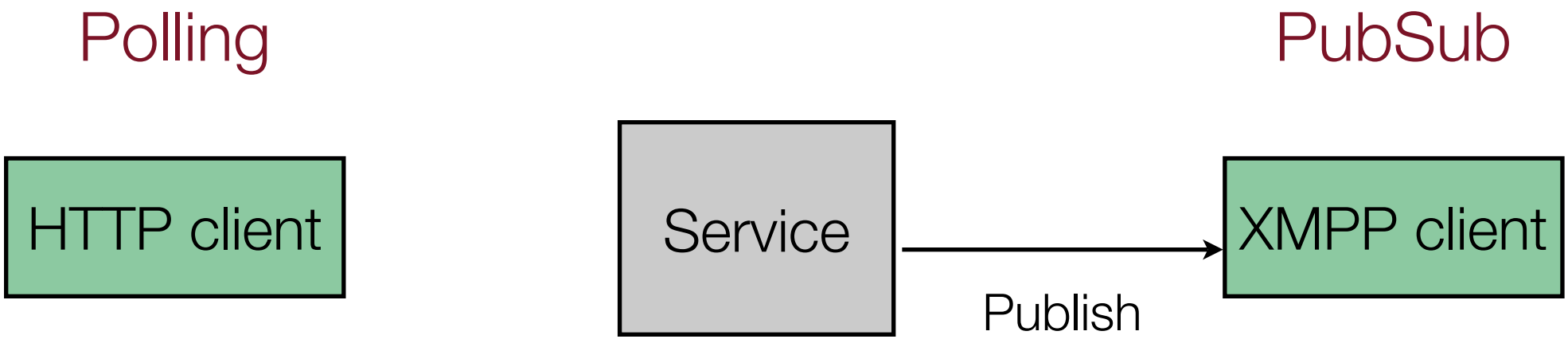
PubSub

HTTP client

Service

XMPP client

PubSub: Polling vs PubSub



PubSub: Polling vs PubSub

Polling

PubSub

HTTP client

Service

XMPP client

PubSub

PubSub

- Protocol to subscribe and publish any kind of information

PubSub

- Protocol to subscribe and publish any kind of information
- Avoids needless polling for new information

PubSub

- Protocol to subscribe and publish any kind of information
- Avoids needless polling for new information
  - Doesn’t scale with the number of users / requests

PubSub

- Protocol to subscribe and publish any kind of information
- Avoids needless polling for new information
  - Doesn’t scale with the number of users / requests
- Extensible: Payload can be any type

PubSub

- Protocol to subscribe and publish any kind of information
- Avoids needless polling for new information
  - Doesn’t scale with the number of users / requests
- Extensible: Payload can be any type
- Configurable

PubSub: Subscribing

```
<iq from="alice@wonderland.lit/rabbithole"
    id="gh921nx3"
    to="notify.wonderland.lit"
    type="set">
  <pubsub xmlns="http://jabber.org/protocol/pubsub">
    <subscribe node="queenly_proclamations"
               jid="alice@wonderland.lit"/>
  </pubsub>
</iq>
```

PubSub: Publishing events


```
<iq from="queen@wonderland.lit/croquetlawn"
    id="ma019r58"
    to="notify.wonderland.lit"
    type="set">
  <pubsub xmlns="http://jabber.org/protocol/pubsub">
    <publish node="queenly_proclamations">
      <item>
        <entry xmlns="http://www.w3.org/2005/Atom">
          <title>A new thought</title>
          <summary>Off with their heads!</summary>
          <link rel="alternate" type="text/html"
                href="http://wonderland.lit/1865/"/>
          <id>tag:wonderland.lit,1865:entry-42</id>
          <published>1865-12-13T18:30:02Z</published>
          <updated>1865-12-13T18:30:02Z</updated>
        </entry>
      </item>
    </publish>
  </pubsub>
</iq>
```

PubSub: Receiving events


```
<message from="notify.wonderland.lit" to="alice@wonderland.lit">
  <body>A new thought: off with their heads!</body>
  <event xmlns="http://jabber.org/protocol/pubsub#event">
    <items node="queenly_proclamations"  id="bl38pahu98h">
      <item id="zi2ba967">
        <entry xmlns="http://www.w3.org/2005/Atom">
          <title>A new thought</title>
          <summary>Off with their heads!</summary>
          <link rel="alternate" type="text/html" h
                ref="http://wonderland.lit/1865/"/>
          <id>tag:wonderland.lit,1865:entry-42</id>
          <published>1865-12-13T18:30:02Z</published>
          <updated>1865-12-13T18:30:02Z</updated>
        </entry>
      </item>
    </items>
  </event>
</message>
```

Extended Presence

Extended Presence

- Publish extra information about yourself

Extended Presence

- Publish extra information about yourself
  - What music are you currently listening to?

Extended Presence

- Publish extra information about yourself
  - What music are you currently listening to?
  - Where are you right now?

Extended Presence

- Publish extra information about yourself
  - What music are you currently listening to?
  - Where are you right now?
  - What are you doing?

Extended Presence

- Publish extra information about yourself
  - What music are you currently listening to?
  - Where are you right now?
  - What are you doing?
- Used to be added to regular presence

Extended Presence

- Publish extra information about yourself
  - What music are you currently listening to?
  - Where are you right now?
  - What are you doing?
- Used to be added to regular presence
  - Doesn’t scale

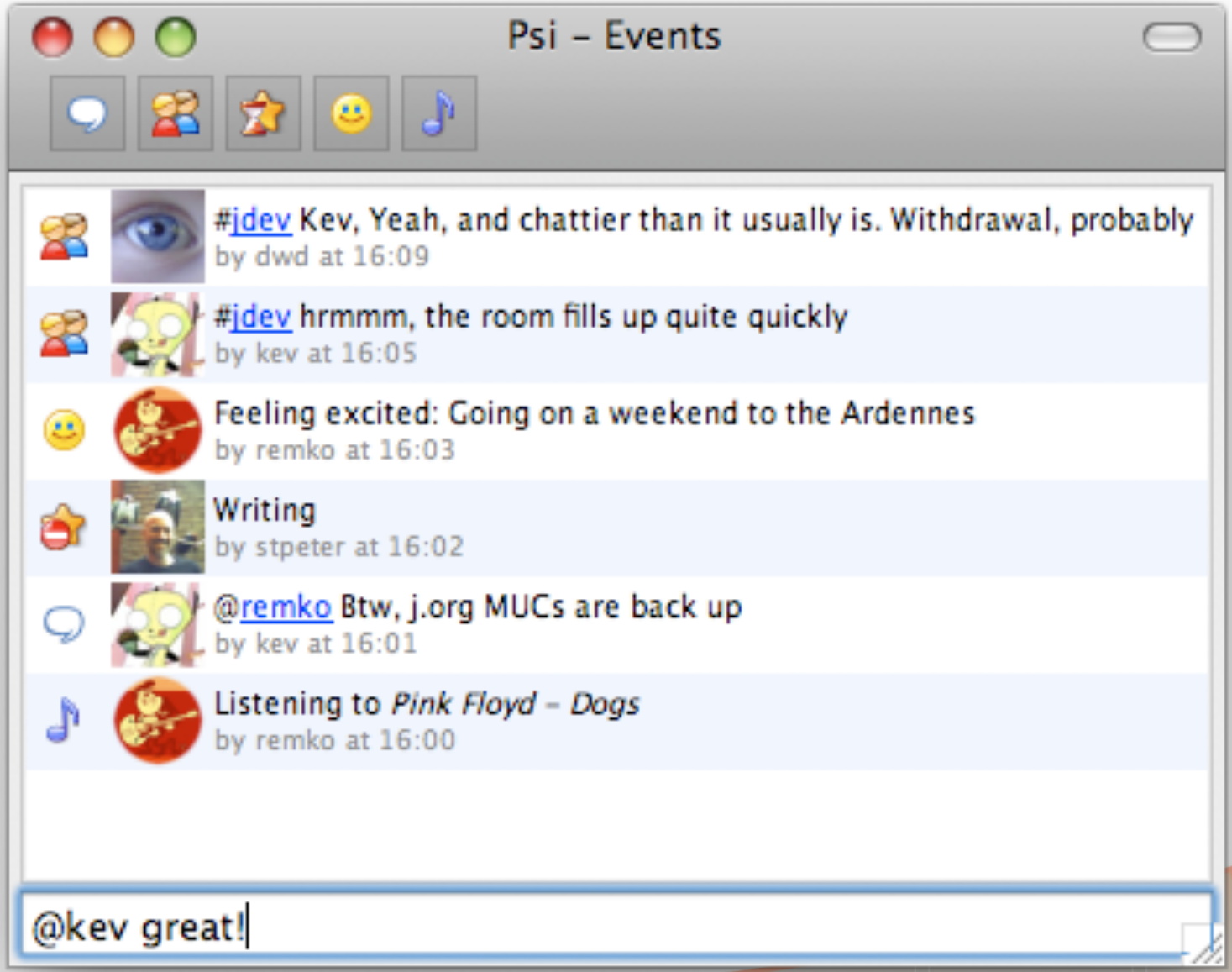
Extended Presence

- Publish extra information about yourself
  - What music are you currently listening to?
  - Where are you right now?
  - What are you doing?
- Used to be added to regular presence
  - Doesn't scale
  - 'Spams' people with unwanted information

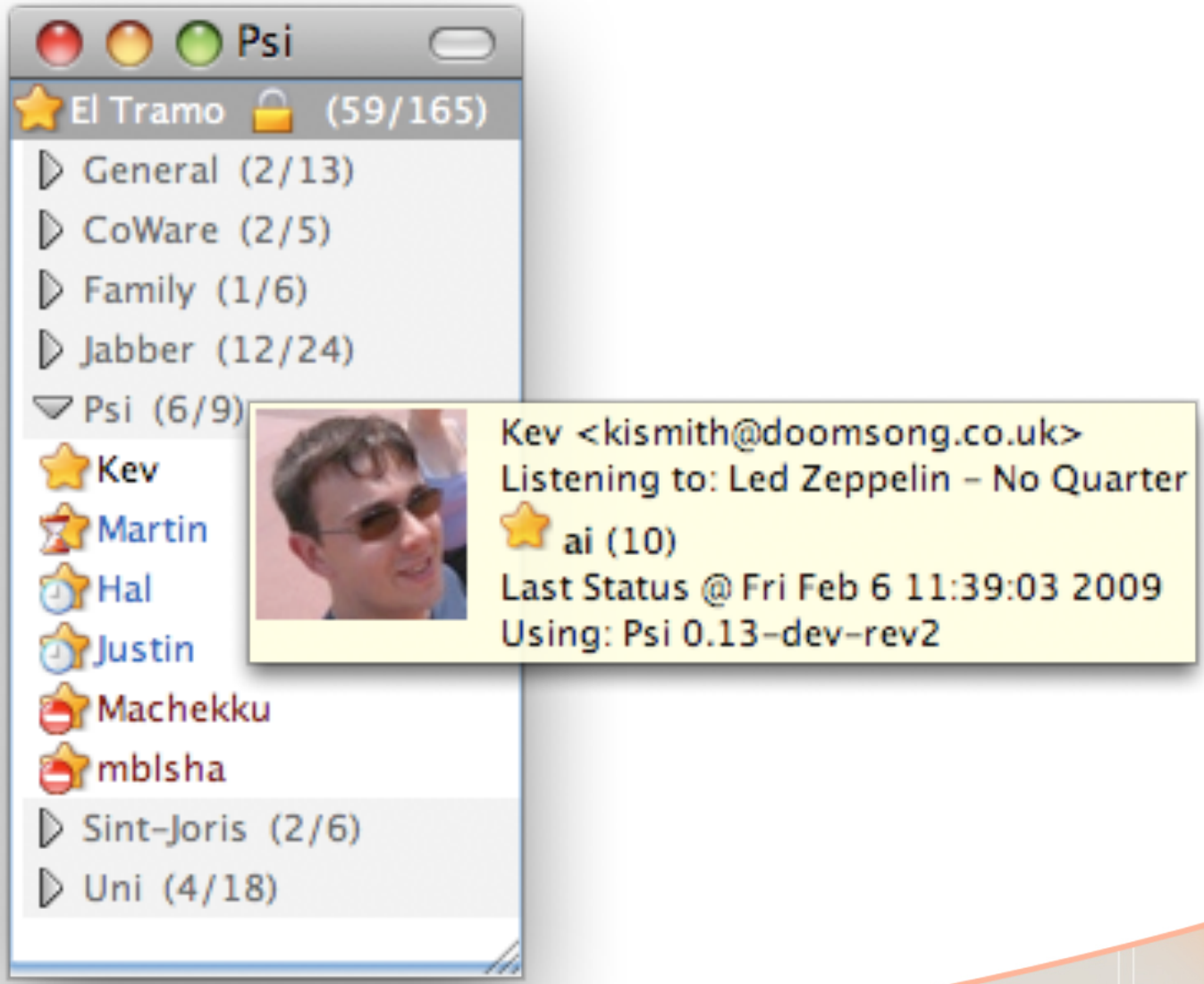
Extended Presence

- Publish extra information about yourself
  - What music are you currently listening to?
  - Where are you right now?
  - What are you doing?
- Used to be added to regular presence
  - Doesn't scale
  - 'Spams' people with unwanted information
- Implemented on top of PubSub

Extended Presence



Extended presence



Jingle

Jingle

- Voice, Video, Multimedia, P2P

Jingle

- Voice, Video, Multimedia, P2P
- XMPP is not optimized to send data streams

Jingle

- Voice, Video, Multimedia, P2P
- XMPP is not optimized to send data streams
- Use XMPP (Jingle) to set up a direct connection between points

Jingle

- Voice, Video, Multimedia, P2P
- XMPP is not optimized to send data streams
- Use XMPP (Jingle) to set up a direct connection between points
  - Negotiate what (Audio/Video) and how (UDP, TCP) you are going to send it

Jingle

- Voice, Video, Multimedia, P2P
- XMPP is not optimized to send data streams
- Use XMPP (Jingle) to set up a direct connection between points
  - Negotiate what (Audio/Video) and how (UDP, TCP) you are going to send it
- Use streaming protocols (RTP,...) to stream the data over the negotiated direct connection

Jingle

- Voice, Video, Multimedia, P2P
- XMPP is not optimized to send data streams
- Use XMPP (Jingle) to set up a direct connection between points
  - Negotiate what (Audio/Video) and how (UDP, TCP) you are going to send it
- Use streaming protocols (RTP,...) to stream the data over the negotiated direct connection
- (Renegotiation during data session)

Jingle

- Voice, Video, Multimedia, P2P
- XMPP is not optimized to send data streams
- Use XMPP (Jingle) to set up a direct connection between points
  - Negotiate what (Audio/Video) and how (UDP, TCP) you are going to send it
- Use streaming protocols (RTP,...) to stream the data over the negotiated direct connection
- (Renegotiation during data session)
- NAT traversal using standard technologies

Jingle

- Voice, Video, Multimedia, P2P
- XMPP is not optimized to send data streams
- Use XMPP (Jingle) to set up a direct connection between points
  - Negotiate what (Audio/Video) and how (UDP, TCP) you are going to send it
- Use streaming protocols (RTP,...) to stream the data over the negotiated direct connection
- (Renegotiation during data session)
- NAT traversal using standard technologies
  - TURN, STUN, ICE

File Transfer

File Transfer

- Various ways of transferring files

File Transfer

- Various ways of transferring files
  - Tiny pieces of data are sent in one stanza

File Transfer

- Various ways of transferring files
  - Tiny pieces of data are sent in one stanza
  - Small pieces of data are sent in a set of consequent <iq/> stanzas

File Transfer

- Various ways of transferring files
  - Tiny pieces of data are sent in one stanza
  - Small pieces of data are sent in a set of consequent <iq/> stanzas
  - Larger pieces of data are sent out of band

File Transfer

- Various ways of transferring files
  - Tiny pieces of data are sent in one stanza
  - Small pieces of data are sent in a set of consequent <iq/> stanzas
  - Larger pieces of data are sent out of band
    - Using SOCKS5

File Transfer

- Various ways of transferring files
  - Tiny pieces of data are sent in one stanza
  - Small pieces of data are sent in a set of consequent <iq/> stanzas
  - Larger pieces of data are sent out of band
    - Using SOCKS5
    - Using older Jingle-like protocol

File Transfer

- Various ways of transferring files
  - Tiny pieces of data are sent in one stanza
  - Small pieces of data are sent in a set of consequent <iq/> stanzas
  - Larger pieces of data are sent out of band
    - Using SOCKS5
    - Using older Jingle-like protocol
    - Being extended to use Jingle

BOSH

BOSH

- Regular XMPP opens one long-lived TCP connection, and keeps it up

BOSH

- Regular XMPP opens one long-lived TCP connection, and keeps it up
- Not always convenient

BOSH

- Regular XMPP opens one long-lived TCP connection, and keeps it up
- Not always convenient
  - Long connections drain batteries on mobile phones

BOSH

- Regular XMPP opens one long-lived TCP connection, and keeps it up
- Not always convenient
  - Long connections drain batteries on mobile phones
  - Web clients cannot keep state (open connections)

BOSH

- Regular XMPP opens one long-lived TCP connection, and keeps it up
- Not always convenient
  - Long connections drain batteries on mobile phones
  - Web clients cannot keep state (open connections)
  - Bad network connectivity

BOSH

- Regular XMPP opens one long-lived TCP connection, and keeps it up
- Not always convenient
  - Long connections drain batteries on mobile phones
  - Web clients cannot keep state (open connections)
  - Bad network connectivity
- Bidirectional streams Over Synchronous HTTP

BOSH

- Regular XMPP opens one long-lived TCP connection, and keeps it up
- Not always convenient
  - Long connections drain batteries on mobile phones
  - Web clients cannot keep state (open connections)
  - Bad network connectivity
- Bidirectional streams Over Synchronous HTTP
  - XMPP over HTTP

BOSH

- Regular XMPP opens one long-lived TCP connection, and keeps it up
- Not always convenient
  - Long connections drain batteries on mobile phones
  - Web clients cannot keep state (open connections)
  - Bad network connectivity
- Bidirectional streams Over Synchronous HTTP
  - XMPP over HTTP
- Cunning mechanism of not having to poll for new messages

BOSH

- Regular XMPP opens one long-lived TCP connection, and keeps it up
- Not always convenient
  - Long connections drain batteries on mobile phones
  - Web clients cannot keep state (open connections)
  - Bad network connectivity
- Bidirectional streams Over Synchronous HTTP
  - XMPP over HTTP
- Cunning mechanism of not having to poll for new messages
  - Piggyback incoming messages on HTTP response

BOSH


```
POST /webclient HTTP/1.1
Host: bosh.wonderland.lit
Content-Type: text/xml; charset=utf-8
Content-Length: 205

<body rid="90029205" sid="3m1ts1htd1s"
      xmlns="http://jabber.org/protocol/httpbind">

  <message to="sister@realworld.lit" xmlns="jabber:client">
    <body>Help, I fell down the rabbit hole!</body>
  </message>

</body>
```

Serverless Messaging

Serverless Messaging

- Sometimes, a server is not available

Serverless Messaging

- Sometimes, a server is not available
  - Remote location without internet

Serverless Messaging

- Sometimes, a server is not available
  - Remote location without internet
  - Conference location with many unknown people you want to communicate wit

Serverless Messaging

- Sometimes, a server is not available
  - Remote location without internet
  - Conference location with many unknown people you want to communicate wit
- Serverless messaging

Serverless Messaging

- Sometimes, a server is not available
  - Remote location without internet
  - Conference location with many unknown people you want to communicate wit
- Serverless messaging
- Uses zero-configuration (Apple) to discover entities on the local network

Serverless Messaging

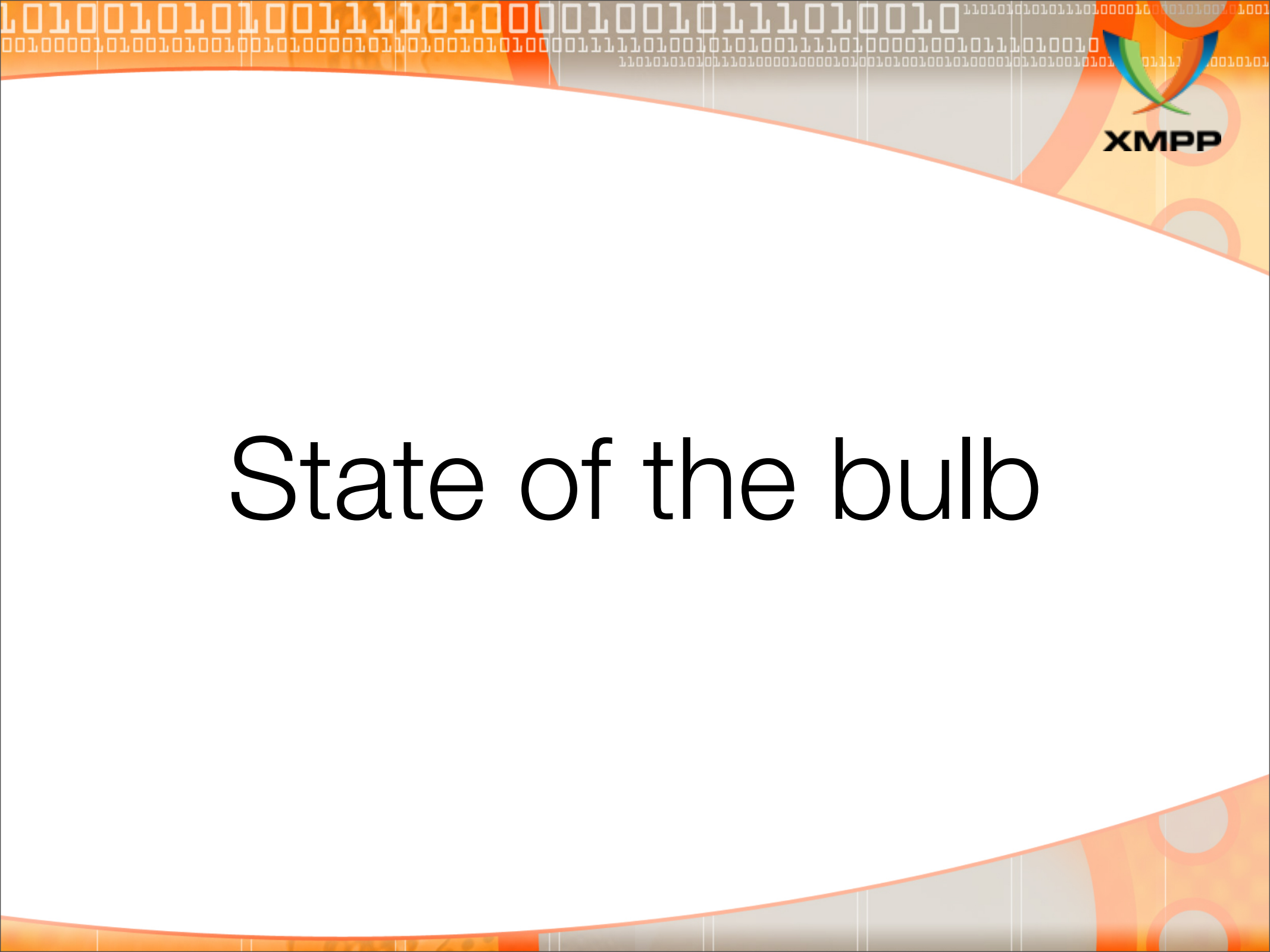
- Sometimes, a server is not available
  - Remote location without internet
  - Conference location with many unknown people you want to communicate wit
- Serverless messaging
- Uses zero-configuration (Apple) to discover entities on the local network
  - mDNS, DNS-SD

Serverless Messaging

- Sometimes, a server is not available
  - Remote location without internet
  - Conference location with many unknown people you want to communicate wit
- Serverless messaging
- Uses zero-configuration (Apple) to discover entities on the local network
  - mDNS, DNS-SD
- Open connection directly to other connection, and send XMPP over that connection

Serverless Messaging

- Sometimes, a server is not available
  - Remote location without internet
  - Conference location with many unknown people you want to communicate wit
- Serverless messaging
- Uses zero-configuration (Apple) to discover entities on the local network
  - mDNS, DNS-SD
- Open connection directly to other connection, and send XMPP over that connection
- iChat's Rendez-vous/Bonjour



State of the bulb

What is the XSF working on?

What is the XSF working on?

- E2E security

What is the XSF working on?

- E2E security
- Whiteboarding

What is the XSF working on?

- E2E security
- Whiteboarding
- File sharing / personal media networks

What is the XSF working on?

- E2E security
- Whiteboarding
- File sharing / personal media networks
- World domination

Conclusion

Join the conversation

- http://**xmpp.org**/
- mailto:**jdev@jabber.org**
- xmpp:**jdev@conference.jabber.org**
- [mailto:|xmpp:]**stpeter@jabber.org**
- [mailto:|xmpp:]**remko@el-tramo.be**
- Jabber/XMPP booth @ FOSDEM
- XMPP Summit on Monday

Conclusion

Conclusion

- The real-time Internet is coming

Conclusion

- The real-time Internet is coming
- Build competitive advantage using open technologies

Conclusion

- The real-time Internet is coming
- Build competitive advantage using open technologies
- What problems can you solve with XMPP?

Conclusion

- The real-time Internet is coming
- Build competitive advantage using open technologies
- What problems can you solve with XMPP?
- Join the conversation

Conclusion

- The real-time Internet is coming
- Build competitive advantage using open technologies
- What problems can you solve with XMPP?
- Join the conversation
- Happy Jabbering!

Conclusion

- The real-time Internet is coming
- Build competitive advantage using open technologies
- What problems can you solve with XMPP?
- Join the conversation
- Happy Jabbering!

More info inside