XMPP/Jabber introducing the lingua franca of instant messaging

Alexander Neumann <fd0@koeln.ccc.de>

27.12.2004

prerequisites

- the protocol I would like to talk about has been named XMPP by the IETF working group. it is specified in RFCs 3920 to 3923. It emerged from a protocol called jabber, and because jabber is easier to pronounce, I will refer to it by the name "jabber" in this talk.

- english is not my native language so please accept my apologies for some small gaps during the talk

purpose of this talk

- give a smooth introduction to the jabber protocol and the more generic xmpp xml routing framework
- everyone should be able to create simple jabber applications (by using a jabber protocol implementation for the language if his choice)
- you should leave this talk with sureness, that instant messaging, using XMPP/Jabber, is simple :)

some facts about jabber

- generic xml routing framework
- paradigm: all the work on the server, clients should be simple to implement
- use server side transports to other systems, not client plugins
- dynamic network of many different servers, not one central server like other IM systems
- can be used in sandboxed environments (like a companies intranet), without connection to the internet

identities in IM systems

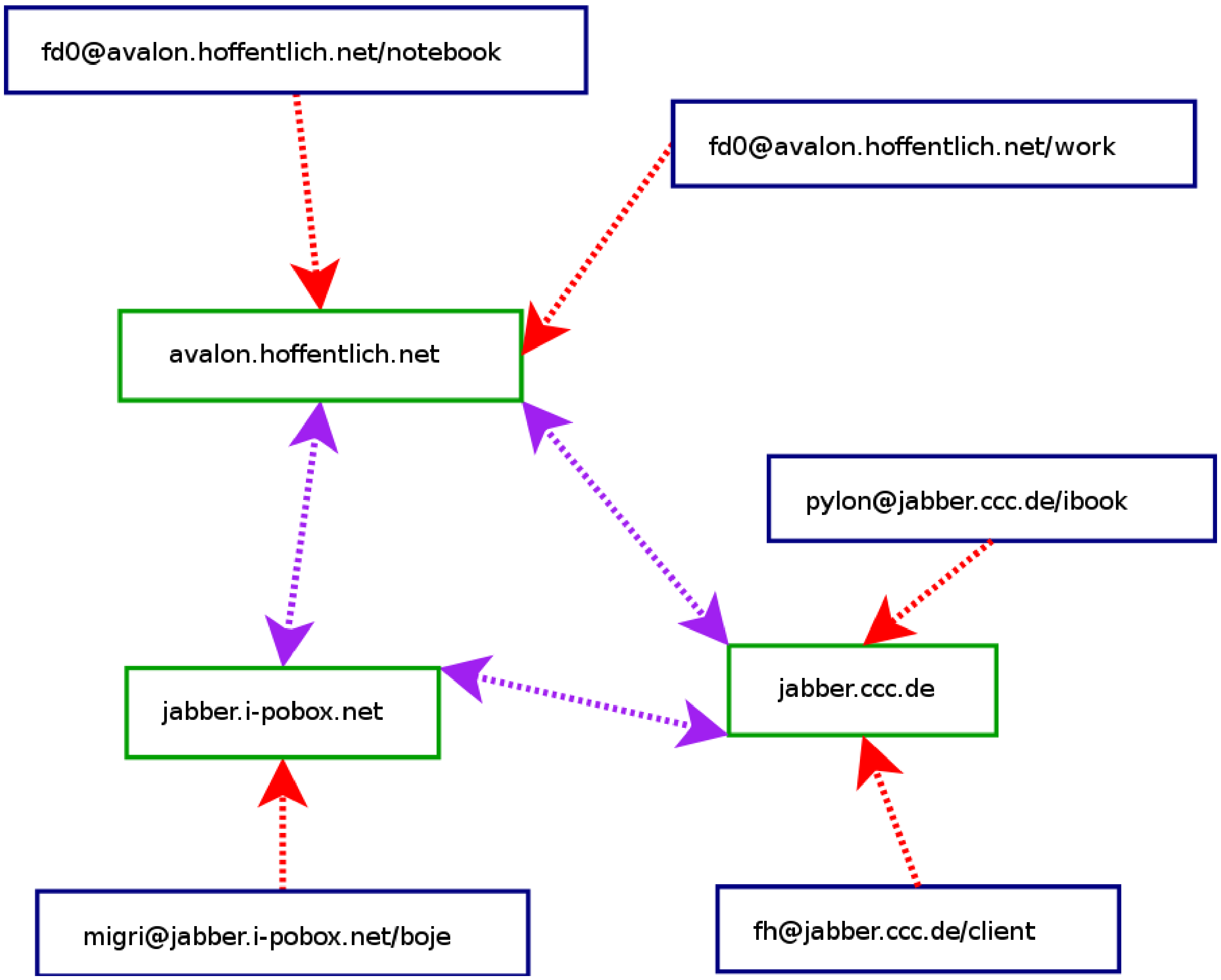
- Identity in ICQ: the UIN.
- Identity in Jabber: the Jabber ID (JID):

  [user@] server.tld [/resource]

Jabber ID

- server.tld:
  - "avalon.hoffentlich.net"
- user@server.tld:
  - "fd0@avalon.hoffentlich.net"
- server.tld/resource
  - "avalon.hoffentlich.net/announce/online"
- user@server.tld/resource
  - "fd0@avalon.hoffentlich.net/necromancer"

jabber network structure example


fd0@avalon.hoffentlich.net/notebook
fd0@avalon.hoffentlich.net/work
avalon.hoffentlich.net
pylon@jabber.ccc.de/ibook
jabber.i-pobox.net
jabber.ccc.de
migri@jabber.i-pobox.net/boje
fh@jabber.ccc.de/client

basic jabber (XMPP) connections, step-by-step

1) create tcp connection
2) establish bidirectional xmlstream
3) (optional) TLS handshake
4) authentication/resource binding
5) message/presence exchange

(specified in RFC 3920)

xmlstreams

- exchange of two complete xml documents (normal connection)
- documents are divided into 'stanzas' (small, well-formed chunks of xml)
- supports asynchronous, event based application design

example for xmlstreams

Jabber client connecting to the server "jabber.org":

```
<?xml version='1.0'?>
<stream:stream to='jabber.org'
    xmlns='jabber:client'
    xmlns:stream="http://etherx.jabber.org/streams'
    version='1.0'>
```

example for xmlstreams #2

```
<?xml version='1.0'?>
<stream:stream from='jabber.org'
    id='someid'
    xmlns='jabber:client'
    xmlns:stream='http://etherx.jabber.org/streams'
    version='1.0'>
```

example for xmlstreams #3

The client terminating the xmlstream:

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

Server responds:

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

valid xml stanzas

valid child-elements of `<stream:stream/>`

- `<presence/>`
- `<message/>`
- `<iq/>`

presence stanza

attributes:

- type: ”available” / ”unavailable”
- to: optional, for presence-stanzas with a specific target JID
- from

children:

- `<show/>`: ”away” /”chat” /”dnd” /”xa”
- `<status/>`: text string, eg ”showering. . . ”

usage of presence stanzas

- exchanging presence information
- managing presence subscription

presence example

Client announces presence to the server:

```
<presence type='available'>
    <status>showering...</status>
</presence>
```

Server distributes information to all authorized contacts.

presence example #2

Client receives presence information from the server:

```
<presence type='unavailable'>
    from='user@server.tld/resource'>
    <status>Disconnected</status>
</presence>
```

message stanza

attributes:

- to: target JID, required
- from: source JID
- type: "normal"/"chat"/"groupchat"/"headline"/. . .

message stanza #2

children:

- `<subject/>`
- `<body/>`
- `<thread/>`
- `<x/>`

message example

Client sends message:

```
<message to='saddam@jabber.gov.iq' type='normal'>
    <subject>WMD?</subject>
    <body>Hey, do you have any WMD?</body>
</message>
```

message example #2

Client receives message:

```
<message to='gwb@whitehouse.gov'
    from='saddam@jabber.gov.iq/ak47'
    type='normal'>
    <subject>Re: WMD?</subject>
    <body>Nop, I don't have any.</body>
</message>
```

iq stanza

attributes:

- type: "get"/"set"/"result"/"error"
- to
- from

children: various

iq example

Client requests server-side filter-list:

```
<iq type='get' id='request1'>
    <query xmlns='jabber:iq:filter'/>
</iq>
```

iq example #2

Server responds:

```
<iq from='user@server.tld' type='result' id='request1'>
    <query xmlns='jabber:iq:filter'>
        <rule>
            <body>moin</body>
            <reply>tachauch</reply>
            <continue/>
        </rule>
    </query>
</iq>
```

Questions?

Thanks for listening (and all the fish)!