Annagram relies on a modified version of the Jupiter Collaboration System, developed at Xerox PARC and extended by Zafer (2001). The algorithm is a distributed operational transformation algorithm.

The system is optimistic in that all local document changes are applied immediately, allowing for reduced latency and immediate user response. It does not rely on a locking measure of any kind. When remote messages are received, it transforms these messages via a transformation function so that the documents are kept in sync. The algorithm tracks the client state and the server state in order to know how to transform the messages.

Annagram utilizes Libjingle for all networking capabilities (chat, VoIP, and Document Sharing). Annagram uses a Managed C++ wrapper that encapsulates Libjingle for access via other .NET languages, such as C#.

Annagram uses XMPP as its protocol for establishing all network sessions. Once a document is connected to, a P2P connection (non-XMPP) with XML messages is used to pass messages.

#### Features
- **Full-featured rich text editor**
- Integration with Google Talk
  - View your contacts and their online status
  - Change your online status
  - Chat with any user using a client that supports Extensible Messaging Presence Protocol (XMPP)
  - Talk with any user using a client that supports the Jingle Audio extension of XMPP.
- **Document Sharing**
  - Share any document with one button.
  - Only a Gmail Name & Document Name combination is needed to join the concurrent editing of any shared document.

#### Architecture

```
|   | C# & .NET | Jupiter Client | Jupiter Server | Libjingle | XMPP XML |
```

#### Jupiter Algorithm

Annagram uses XMPP as its protocol for establishing all network sessions. Once a document is connected to, a P2P connection (non-XMPP) with XML messages is used to pass messages.

### Annagram

Current text editing software have limits to their editing capabilities. Text editing software has traditionally allowed for only one user to be editing a document at a time. As users needs and desires for features have adapted to a networked and multi-user world, the experience of text editing has remained a single user experience. It is easy to imagine situations in which users would benefit from a multi-user editing environment.

Examples of situations include students working on a group project, developers who desire to work on the same source files, and corporate users operating within a team-environment to compile a team-wide report.

Annagram allows for concurrent editing of a document by multiple users, i.e. two users (or more) can edit one document at the same time. Annagram is a concurrent text editor for the Windows platform that will facilitate group collaboration through the means of multiple users being allowed to take part in the editing process and other modes of communication (i.e. VoIP & chat).

Annagram utilizes Libjingle for all networking capabilities (chat, VoIP, and Document Sharing). Annagram uses a Managed C++ wrapper that encapsulates Libjingle for access via other .NET languages, such as C#.

Annagram uses XMPP as its protocol for establishing all network sessions. Once a document is connected to, a P2P connection (non-XMPP) with XML messages is used to pass messages.

### Features

- **Full-featured rich text editor**
- Integration with Google Talk
  - View your contacts and their online status
  - Change your online status
  - Chat with any user using a client that supports Extensible Messaging Presence Protocol (XMPP)
  - Talk with any user using a client that supports the Jingle Audio extension of XMPP.
- **Document Sharing**
  - Share any document with one button.
  - Only a Gmail Name & Document Name combination is needed to join the concurrent editing of any shared document.

### Architectural Components

```
|   | C# & .NET | Jupiter Client | Jupiter Server | Libjingle | XMPP XML |
```

### Jupiter Algorithm

Annagram relies on a modified version of the Jupiter Collaboration System, developed at Xerox PARC and extended by Zafer (2001). The algorithm is a distributed operational transformation algorithm.

The system is optimistic in that all local document changes are applied immediately, allowing for reduced latency and immediate user response. It does not rely on a locking measure of any kind. When remote messages are received, it transforms these messages via a transformation function so that the documents are kept in sync. The algorithm tracks the client state and the server state in order to know how to transform the messages.

The Jupiter Algorithm relies on 2-dimensional state spaces, ensuring that all clients will reach the same state (although most likely via different paths). An example is below:

#### Initial String

```
|   | 0, 0 |
```

#### Final String

```
|   | 0, 2 |
```

---

Senior Project Poster Day 2006, CIS Dept., University of Pennsylvania