GaggleBridge: collaborative data analysis

Florian Battke*†, Stephan Symons†, Alexander Herbig and Kay Nieselt

ABSTRACT

**Motivation:** Tools aiding in collaborative data analysis are becoming ever more important as researchers work together over long distances. We present an extension to the Gaggle framework, which has been widely adopted as a tool to enable data exchange between different analysis programs on one computer.

**Results:** Our program, GaggleBridge, transparently extends this functionality to allow data exchange between Gaggle users at different geographic locations using network communication. GaggleBridge can automatically set up SSH tunnels to traverse firewalls while adding some security features to the Gaggle communication.

**Availability:** GaggleBridge is available as open-source software implemented in the Java language at [http://it.inf.uni-tuebingen.de/gb](http://it.inf.uni-tuebingen.de/gb).

**Contact:** florian.battke@uni-tuebingen.de

Received on May 10, 2011; revised on June 28, 2011; accepted on July 14, 2011

1 INTRODUCTION

The Gaggle (Shannon et al., 2006) has become the standard framework for connecting different applications together to study systems biology questions more efficiently. Many different applications support Gaggle, i.e., they are implementing the Gaggle Goose interface. Data such as gene identifiers, clusters, regulatory applications support connecting multiple Gaggle Boss instances, thus creating a sort of ‘Super-Boss’ instance (Fig. 1A). The elegance of this approach is that it requires neither modification of the existing Gaggle Boss implementation nor of any Goose. Instead, GaggleBridge represents itself as just another Goose to each of the connected Bosses, transferring data between them. Connected Boss instances can be activated or inactivated, similar to the possibility of activating and deactivating connected Geese in the Boss main window. However, while inactivated Geese can still send data to their Boss, deactivation in GaggleBridge completely disconnects the respective Boss such that it can neither send nor receive from other Boss instances connected to the GaggleBridge.

Since any academic users are behind restrictive firewalls, and also do not want unauthorized users to connect to their Gaggle Boss, we decided to include SSH tunneling in GaggleBridge. Thus, a group of researchers can collaborate as long as at least one of their computers is reachable from the outside via SSH login (as in Fig. 2C). Many research facilities permit SSH traffic to pass their firewalls, effectively enabling users to connect otherwise secluded Gaggle Boss instances together. Our implementation is flexible enough to support many different network configurations (see Fig. 2 for some examples), allowing users in extremely restrictive networks, or behind NAT (network address translation) routers to partake in the collaborative data analysis. Since all participating Gaggle Boss instances are equal peers (being both servers and clients at the same time), a dedicated machine allowing users to login via SSH can be used to set up a Gaggle network in a star topology. Thus, only one computer needs to be set up to allow SSH logins and administration can be centralized.

The SSH protocol encrypts all data before transmission, using a very secure encryption method. Data communicated between the Gaggle users thus is protected from eavesdropping. Furthermore, since each user needs SSH login credentials (username and password) to connect, this adds an admittedly simple form of user management to the Gaggle. We have added SSH tunneling capabilities to GaggleBridge based on the JSch implementation ([http://www.jcraft.com/jsch/](http://www.jcraft.com/jsch/), licensed under a BSD-style open-source license) of the SSH2 protocol.

Furthermore, we developed an extended version of the Gaggle Boss that informs the user about Goose that try to connect and allows the user to refuse unwanted Goose before they can establish a connection. If a Goose is denied, it will neither receive any objects transmitted nor will it be able to send data to other Goose on this Boss. Both the extended Boss and GaggleBridge can be used in one single application. Alternatively, GaggleBridge can be used as a stand-alone program.

---

*To whom correspondence should be addressed.
†The authors wish it to be known that, in their opinion, the first two authors should be regarded as joint First authors.
Fig. 1. (A) screenshot showing GaggleBridge with three connected Boss instances, one connection to the locally running Boss instance, one connection to a remote instance without tunneling (user Alex) and one remote connection via SSH tunneling (user Lucía). (B) screenshot of a Gaggle Genome Browser instance (user Lucía) broadcasting a selection of three loci; (C) screenshot of a Gaggle Genome Browser instance (user Alex) receiving this broadcast. The loci are highlighted and added to list of bookmarks on the right.

3 DISCUSSION

The availability of automated network communication, which is provided by GaggleBridge, results in a wide range of possible applications. While the main focus of Gaggle is data exchange between different bioinformatic tools, the connection of several instances of the same tool running on different computers now enables collaborating scientists at different locations to work simultaneously on the same data.

Using Gaggle in a networked environment has two inherent security problems: first, in an open network configuration, anyone can connect to a running Gaggle Boss, whether the original user wants this or not. Secondly, when using GaggleBridge, any of the connected users could allow a third party to take part in the information transmitted, either locally or by connecting them with a further GaggleBridge connection. While our enhanced Boss somewhat remedies the first problem, we believe that this security issue should be addressed in a possible later release of the Gaggle itself. The second problem is the general problem of trust between collaborating researchers. Currently, we cannot propose a technical solution for this issue.

To illustrate the benefit of our approach, we present the example of two users (Lucía and Alex) working in two different labs on the same project. Both use the Gaggle Genome Browser (Bare et al., 2010) to visualize locus data, in addition they have a conversation via a telephone conference. Their Gaggle Boss instances are connected via GaggleBridge. Now Lucía can select specific loci in the browser and broadcast the selection with Gaggle (Fig. 1B). Alex’ genome browser receives this information, the same elements are instantly selected and the loci are added to the list of bookmarks (Fig. 1C). This increases the efficiency of the communication between the users substantially as a verbal description of the locus by Lucía and a tedious manual search for the elements by Alex is no longer required. Such communication can also take place within a much larger group of users, and each user can run additional Geese, which will all receive the broadcasts processing and visualizing the data according to their specific functionality.

Note that, as GaggleBridge transparently enhances Gaggle, no modification of these Geese or the connected Boss instances are required. Thus, the features of GaggleBridge can be used with minimal installation effort, allowing to swiftly introduce a collaborative approach to data analysis into any research environment.

ACKNOWLEDGEMENT

The authors are grateful to Lucía Spangenberg-Torre for providing a transatlantic test environment.

Funding: Florian Battke was supported by the DFG Priority Program 1335 ‘Scalable Visual Analytics’, Alexander Herbig by grants of the ERA-NET SysMO Project (GEN2006-27745-E/ZYS).

Conflict of Interest: none declared.

REFERENCES
