



# Swirl Project

visualization for effective security

## Motivation

Building systems that are both secure and usable is a persistent challenge. We need to ensure that information can be both protected and shared. Participation in online information exchanges (commercial, personal, and civil) requires that people be able to trust the infrastructures that support these exchanges. Advances in security and cryptography technologies has provided tools that enhance the *theoretical* security of information systems -- that is, the level of secure communication and computation that is technically achievable. However, *effective* security depends not only on the technology of security, but on its application. An unusable security system is as good as no security at all.

One approach is to make security transparent – to embed it in the system at a low enough level that users need never be aware or concerned with it. However, we believe that the variety of user needs and usage settings is so large that decisions about security cannot be hidden. End users do not simply need security, but rather need *the ability to make decisions about their security needs*. Informed decision making matches the capabilities of the system to the needs of the moment. Empirical investigations of end-user behavior with networked and distributed systems suggests that this decision-making process poses many practical problems.

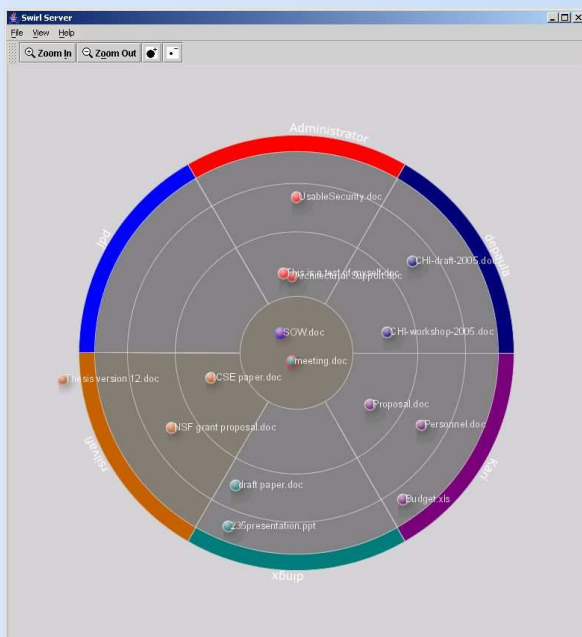
Our goal is to create a technical infrastructure which makes visible the configuration, activity, and implications of available security mechanisms, thereby allowing end users to make informed security choices resulting in increased *effective* security.

## Approach

We are exploring three design principles in order to further refine our theoretical approach:

- **Visualization mechanisms:** By providing users with dynamic feedback on relevant but hidden aspects of system activity, our goal is to provide people with a means to understand the relationship between their actions and the technology configuration through which they are performed.
- **Integration of configuration and action:** Conventional interfaces separate configuration and action both space and time, usually providing a separate control panel in which configuration is performed. Our design approach seeks to make configuration and action part of the same interactional space.
- **Event-based architectures:** These architectures route, combine, and process information from many different system elements. This event-based approach allows systems to be both extensible and scalable, across devices and across platforms.

## The Impromptu File Sharing Application



We have developed *Impromptu*, an ad-hoc peer-to-peer file sharing application, as a test bed to explore these design principles and to measure the effectiveness of our approach.

- Each slice represents a user's workspace. These slices expand and contract as users join and leave a session.
- Dots represent shared files that are currently being shared within the workspace
- Color markers on the perimeter of each slice uniquely identify each user.
- Dot color indicates the file owner.
- Colored rings around each dot designate who last accessed a file. These rings blink for a brief period of time following an access event on the file.
- Multiple concentric regions indicate different sharing levels, with increased sharing permissions as files are dragged toward the center of the "pie".
- A file placed in center space will persist within a session even after its owner has exited. When a user exits, all non persistent files disappear along with the user's pie wedge.

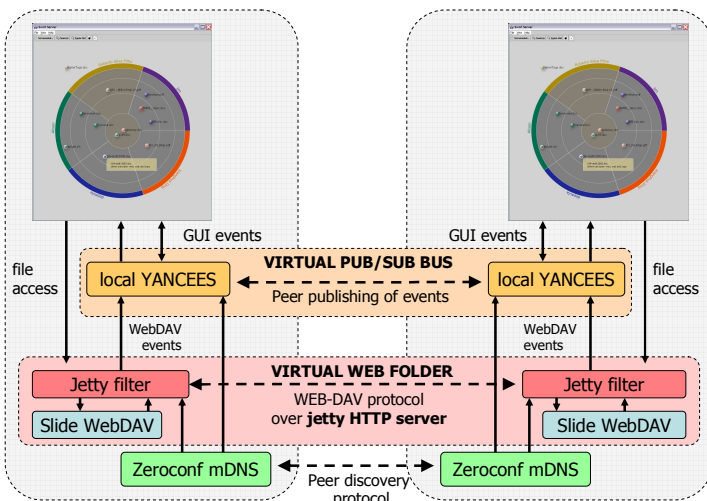
## Impromptu Requirements

Impromptu is designed to support face-to-face, ad-hoc file sharing. The goals of the testbed imply four significant constraints on software design and implementation.

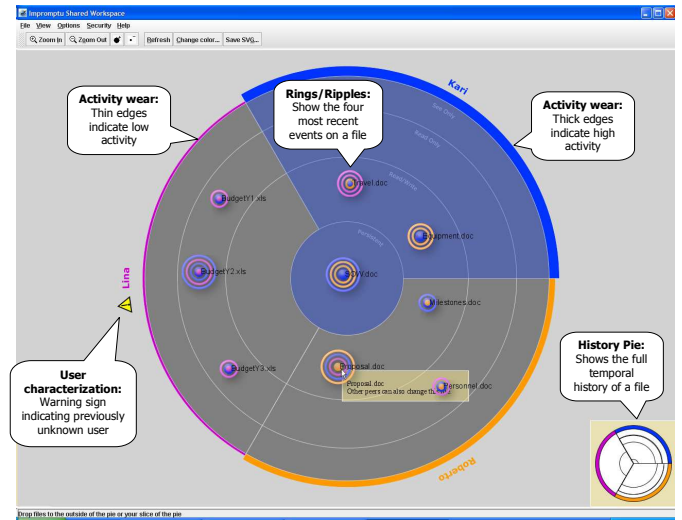
- **Zero configuration:** the overhead for setting up a collaborative file space should be negligible.
- **No pre-registration:** since sharing is ad hoc, it should require no prior registration of relevant parties.
- **No fixed infrastructure:** ideally, the system should be operable with no fixed infrastructure; it should not require, for example, connection to the public Internet.
- **Cross-platform:** it should operate on a wide number of platforms.

## Impromptu Implementation

The current Impromptu architecture is illustrated above. Each client's files are stored in a WebDAV repository. WebDAV provides a standard interface to access files and control access permissions. Each client runs a local instance of Jetty, a Java HTTP server containing a Slide WebDAV servlet. A Jetty filter stitches these separate servers/repositories together to create on each client a unified, virtual shared space. WebDAV was chosen for its easy integration with current operating systems being broadly accessible across platforms both through Web interfaces and also through native file system interfaces on a range of systems including Windows, MacOS X, and Linux. Peer discovery is accomplished using an implementation of the IETF Zeroconf protocols. Impromptu uses event-based integration. Events are used both to visualize dynamic activity as well as to ensure view consistency. The event bus was implemented using the YANCEES configurable and extensible event service.



The Impromptu architecture is an event-based infrastructure, based on YANCEES, that integrates peer-to-peer (P2P) and WEBDAV file sharing protocols into a unified repository model.



This screenshot of the impromptu application highlights new visualizations for history, temporal consistency and user characterization.

## New Visualizations

We have been developing a number of visualization extensions to the framework based on feedback from various user studies. These extensions primarily focus on history and temporal consistency.

**Rings and Ripples** – Concentric rings now “ripple out” from the document dot icon. These rings indicate the four most recent activities on the file. The most recent activity is attached to the dot itself, older events radiate outwards and eventually disappear.

**History Pie** – Provides a complete temporal description of all activities on a file over the full duration of a session. Mousing over a dot displays a history pie in the lower right corner of the user interface. The arcs within a wedge correspond to a particular user's activity on that file. The timeline stretches from the current time at the center to the start of the session at the pie's outer edge.

**Activity Wear** – The edge of each pie slice displays an indication of the accumulated history of an individual's action. Each user's activity is calculated relative to the other users' activities. A maximum width border indicates a very active user and a thin minimum width indicates a relatively inactive user.

**User Characterization** – A warning triangle indicates that a particular user may not have participated in a session with this user before. It indicates that either the username is unknown or that there is an unexpected mapping between the user's name and their hardware ethernet address.

### Contact Information

Professor Paul Dourish  
 Professor David F. Redmiles  
 Institute for Software Research  
 University of California  
 Irvine, California 92697-3425  
 {redmiles, jpd}@ics.uci.edu  
 +1-949-824-{8127, 3823}  
 Fax 949-824-1715

To learn more about the Swirl project, please visit the website:

<http://isr.uci.edu/projects/swirl/>

This material is based upon work sponsored by the National Science Foundation under grant numbers 0326105 and 0534775 and by the Intel Corporation. The content of the work should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of either organization.

June 2006