Undo: Update and Futures

Aaron Brown

ROC Research Group
University of California, Berkeley

Summer 2003 ROC Retreat
5 June 2003

RECOVERY-ORIENTED COMPUTING

Outline

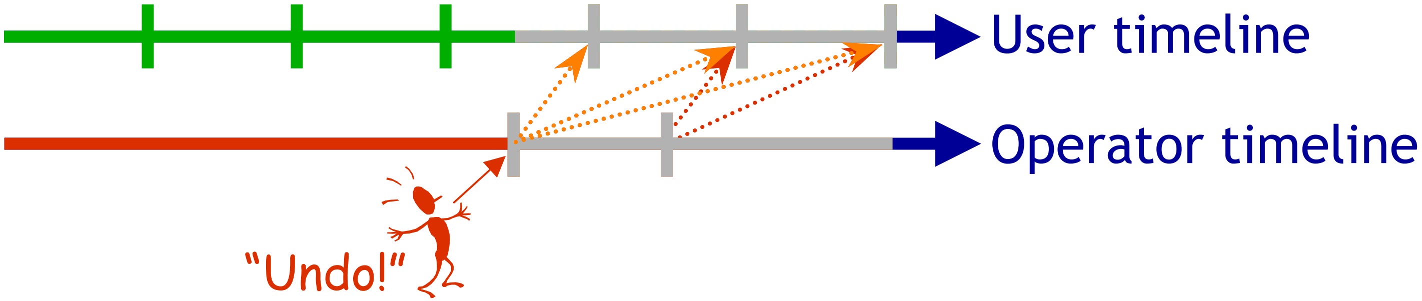
- Recap of Undo for Operators
- Measurements of e-mail undo prototype
- Upcoming: human evaluation
- Potential future extensions

Recap: What Is “Operator Undo”?

- **Give operators and system admins the ability to “travel in time”**
  - – to undo the effects of erroneous actions
    - » configuration changes
    - » new software deployment
    - » patches and upgrades
    - » problem repairs
  - – to retroactively repair other problems affecting state
    - » software bugs
    - » viruses
    - » external attacks

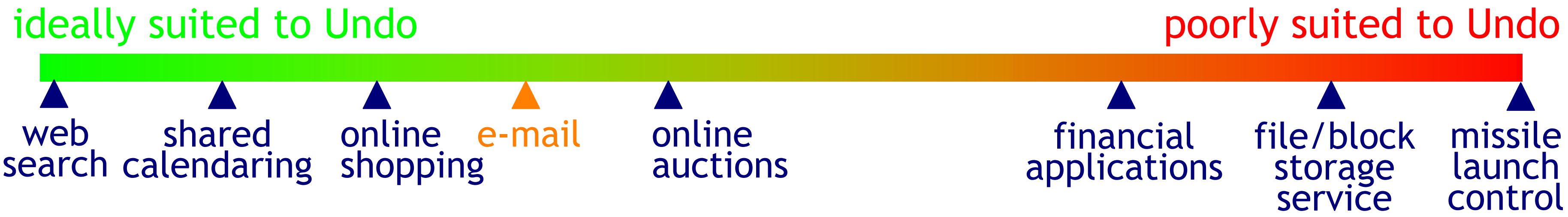
Recap: Three R's Undo Model

- **Time travel for system operators**
  - **Rewind:** roll back all state, users' and operator's
  - **Repair:** alter past operator events to avert problems
  - **Replay:** re-execute rewound user events
    - » operator timeline must be restored manually, if desired
    - » may cause externally-visible paradoxes for users



A Simple Solution for a Common Case

- **Undo for services with human end-users**
  - – centralized state scopes the problem
  - – human users provide flexibility for handling paradoxes
    - » undo is typically transparent to end-user, but not perfect
    - » worst-case: end-user must reconcile mental model based on supplied hints
- **Applicability**



ideally suited to Undo

poorly suited to Undo

web search

shared calendaring

online shopping

e-mail

online auctions

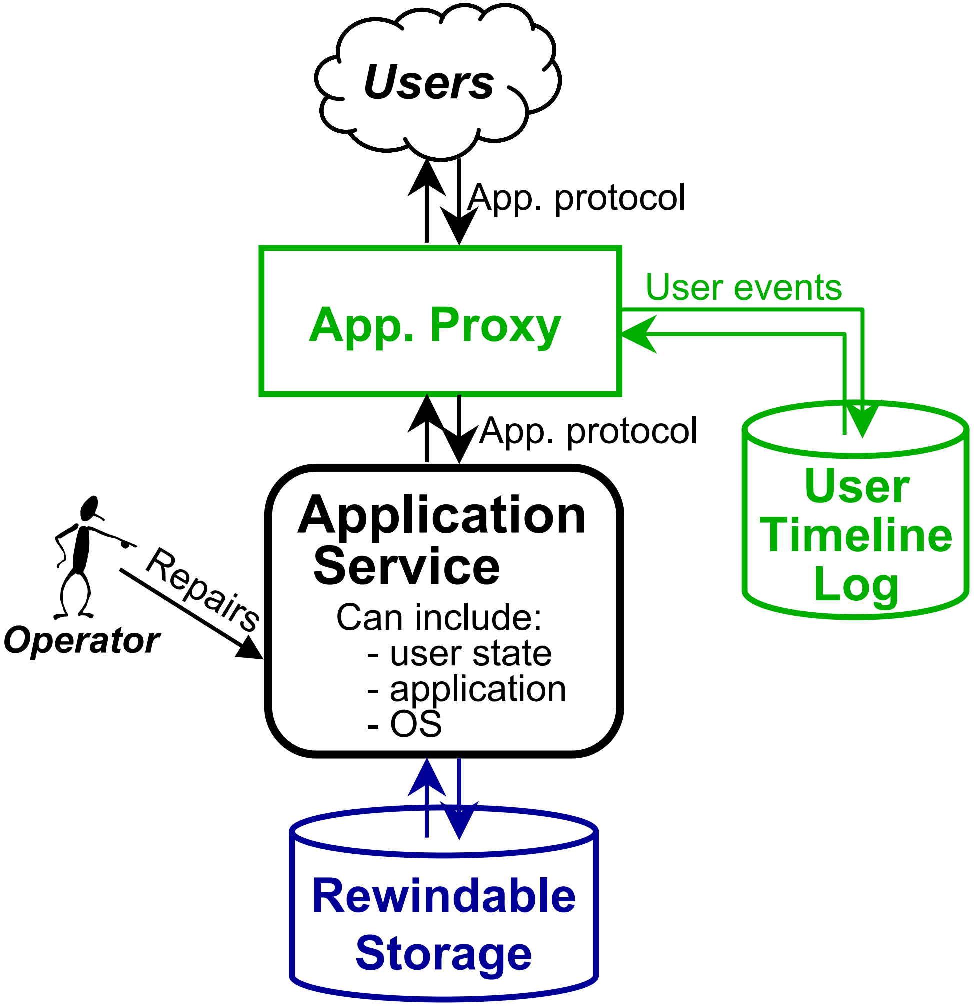
financial applications

file/block storage service

missile launch control

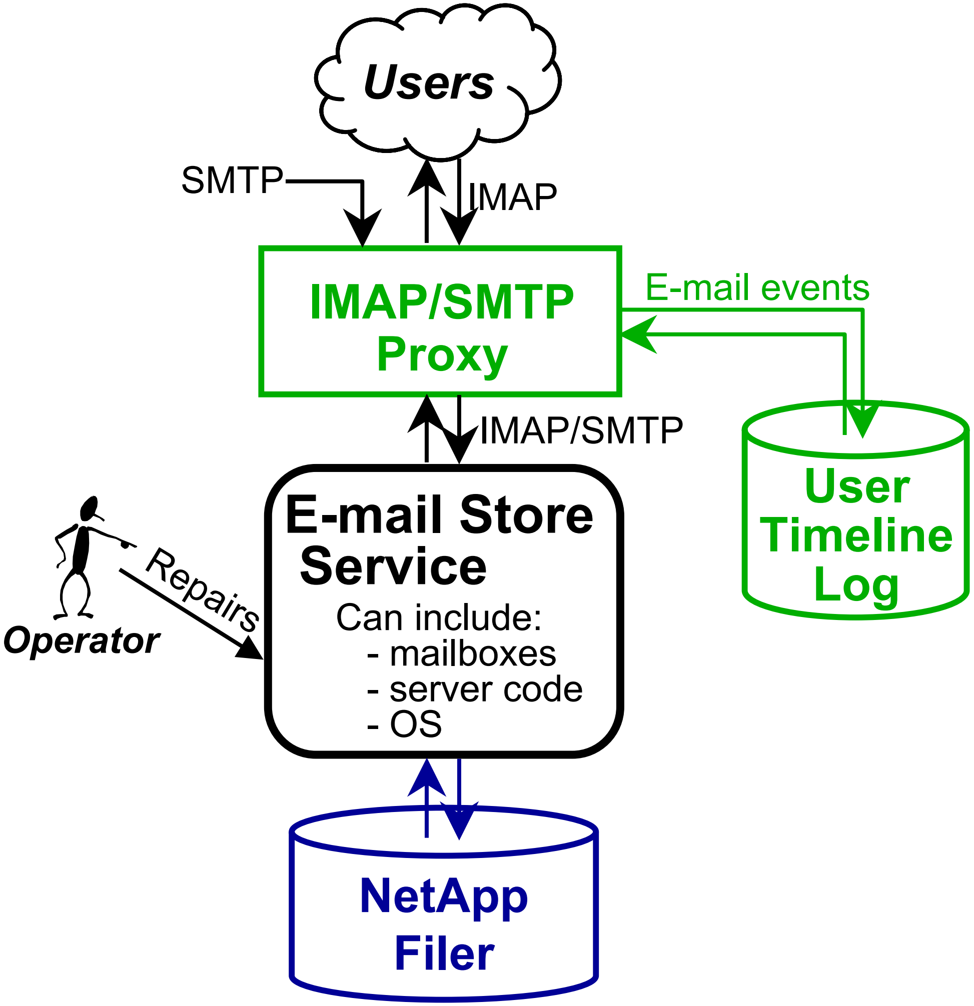
Architecture in Brief

- **Target**
  - black-box services with human end-users
  - single-host, for simplicity
- **Approach**
  - rewindable storage
  - intercept, log, replay user requests
- **Fault assumptions**
  - service can be arbitrarily incorrect



Instantiation: E-mail Prototype

- **Prototype target**
  - e-mail store service
    - » leaf node in e-mail delivery network
- **Implementation**
  - NetApp filer provides rewindable storage layer
  - e-mail-specific proxy intercepts/replays IMAP & SMTP requests



Key Concept: Verbs

- **Verbs encode user events**
  - encapsulate application protocol commands
    - » record of desired user action
    - » context-independent record of parameters
    - » record of externally-visible output
  - intended to capture intent of protocol commands, not effects on system state
- **Example verbs for e-mail** (simplified)
  - **SMTP:** DELIVER {to, from, messageText} {}
  - **IMAP:** COPY {srcFolder, msgNum[], dstFolder} {}
    FETCH {folder, msgNum[], fetchSpec} {*text*}

Role of Verbs

- **Verbs enable replay**
  - verb log forms a history of end-user interaction
    - » dissociated from original system context
    - » annotated with original output to end-user
    - » annotated with external consistency policy and compensations for consistency violations
- **Verbs make it easier to reason about 3R's**
  - define exactly what user state is preserved by 3R cycle
- **Verbs capture key application semantics**
  - consistency model and commutativity of operations

Outline

- Recap of Undo for Operators
- Measurements of e-mail undo prototype
- Upcoming: human evaluation
- Potential future extensions

E-mail Prototype Details

- **Target service: e-mail store service**
  - – a leaf node in the Internet e-mail network
- **Prototype details**
  - – wraps an existing IMAP/SMTP e-mail store service
    - » not platform-specific
    - » evaluation uses sendmail and the UW IMAP server
  - – written in Java
    - » ~25K lines (~9K semicolons)
    - » about 1/8 the size of the mail service itself, in LoC

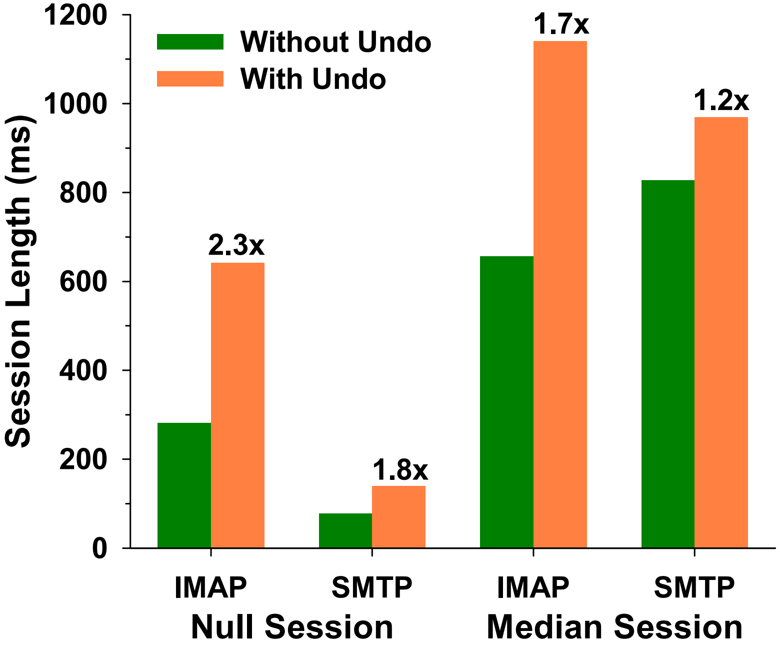
Prototype Measurements

- **Experiments**
  - space overhead
  - time overhead
  - rewind & replay time
- **Evaluation workload**
  - modified SPECmail2000 workload with 10,000 users
    - » simulates traffic seen by ISP mail server
    - » modified to use IMAP instead of POP; all mail kept local

Feasibility: Space & Time Overhead

- **Space overhead**
  - 0.45 GB/day/1000 users
    - » uncompressed
    - » Java serialization bug overhead factored out (>2x bigger)
  - ~250,000 user-days of data on one 120GB disk

- **Time overhead**
  - IMAP/SMTP session lengths for SPECmail workload:

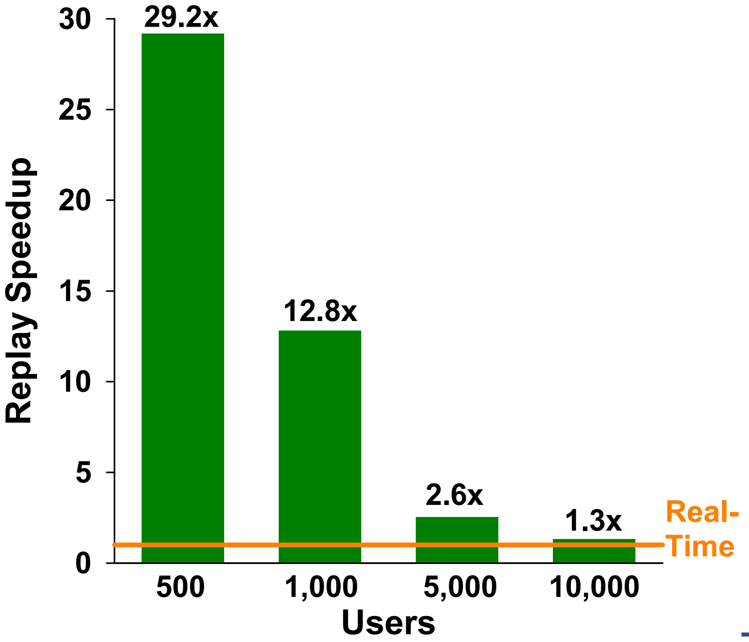


  - below perceived "sluggishness" threshold for interactive apps.

Feasibility: Rewind and Replay

- **Rewind**
  - NetApp filer snapshot restore: ~8 seconds
    - » independent of amount of data to restore
    - » but not undoable
  - alternative is O(#files)
    - » 10 minutes for 10,000 users

- **Replay**
  - replay speed: ~9 verbs/sec
  - with parallel, O-O-O replay
  - better connection management will help
  - compared to real-time:



Outline

- Recap of Undo for Operators
- Measurements of e-mail undo prototype
- Upcoming: human evaluation
- Potential future extensions

Evaluating Undo: Human Factors

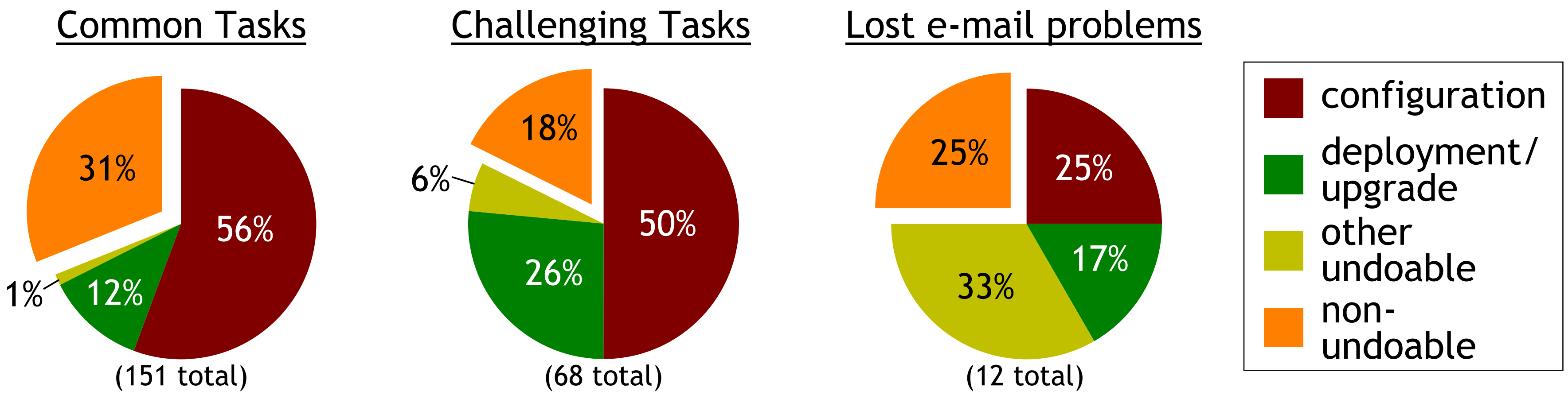
- Undo is a recovery tool for <u>human operators</u>
  - effectiveness depends on how it is used
    - » will it address the problems faced by real operators?
    - » will operators know when/how to use it?
    - » does it improve dependability over manual recovery?
- Need methodology that synthesizes systems benchmarking with human studies
  - include human operators to drive recovery
  - but focus is on the system and system metrics
    - » recovery time, dependability, performance

Evaluating Human Factors of Undo

- **Three-step process**
  1) survey operators to identify real-world problems
     - » evaluate whether Undo will address them
     - » collect scenarios for step 2
  2) controlled laboratory experiments involving humans
     - » evaluate Undo against manual recovery
     - » use scenarios from step 1
     - » evaluate with dependability metrics: recovery time, correctness, performance
  3) long-term ethnographic study of deployed system
     - » evaluate dependability benefits of Undo "in the wild"
     - » requires time and resources beyond the scope of this work

Step 1: Survey Operators

- **Online survey of e-mail system operators**
  - – questions on daily tasks, challenges, recent problems
  - – 68 responses
- **Results**



» configuration and deployment issues dominate

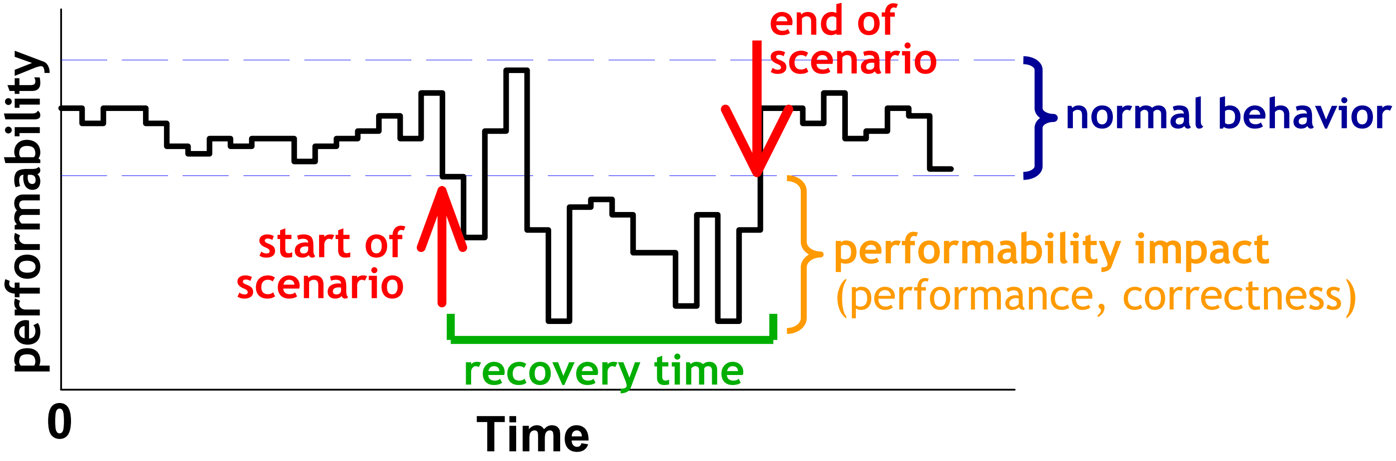
» Undo potentially useful for majority of tasks, problems

Step 2: Lab Experiments w/Humans

- Questions to answer
  - do operators know when Undo is appropriate?
  - does having Undo improve dependability?
- Compare e-mail systems with & without Undo
  - randomized human trials
  - each trial structured as a dependability benchmark
- In progress

Dependability Benchmarks

- **Dependability benchmark basics**
  - apply workload
  - simulate realistic problem scenario
  - measure recovery time, correctness, performance



  - trial scenarios chosen based on survey results
    - » including scenarios where Undo is unlikely to help

See: Brown, Chung, Patterson, "Including the Human Factor in Dependability Benchmarks", *DSN WDB* 2003.
Brown, Patterson, "Towards Availability Benchmarks...", USENIX 2000.

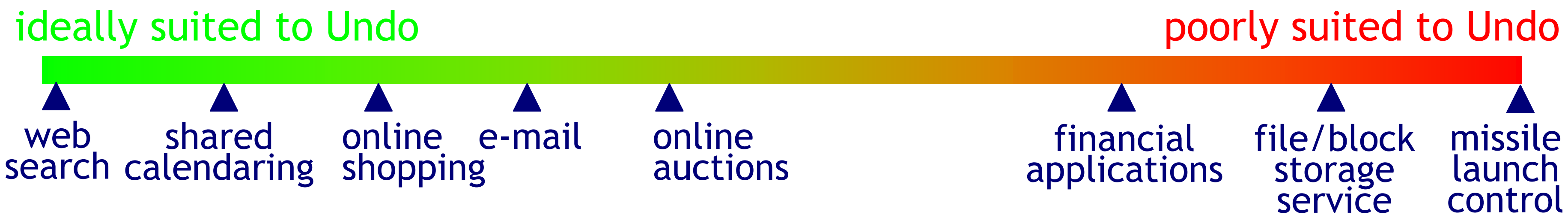
Lab Experiments with Humans

- **Some key subtleties**
  - overcoming mental model inertia
    - » select and train less-experienced subjects
  - making scenarios tractable
    - » subject plays role of shift-work operator repairing documented problem from previous shift
- **Status: in progress**
  - experimental protocol defined
  - just received Human Subjects Committee approval
  - data collection to begin shortly

Outline

- Recap of Undo for Operators
- Measurements of e-mail undo prototype
- Upcoming: human evaluation
- Potential future extensions

Extending Undo: Other Apps



ideally suited to Undo

poorly suited to Undo

web search

shared calendaring

online shopping

e-mail

online auctions

financial applications

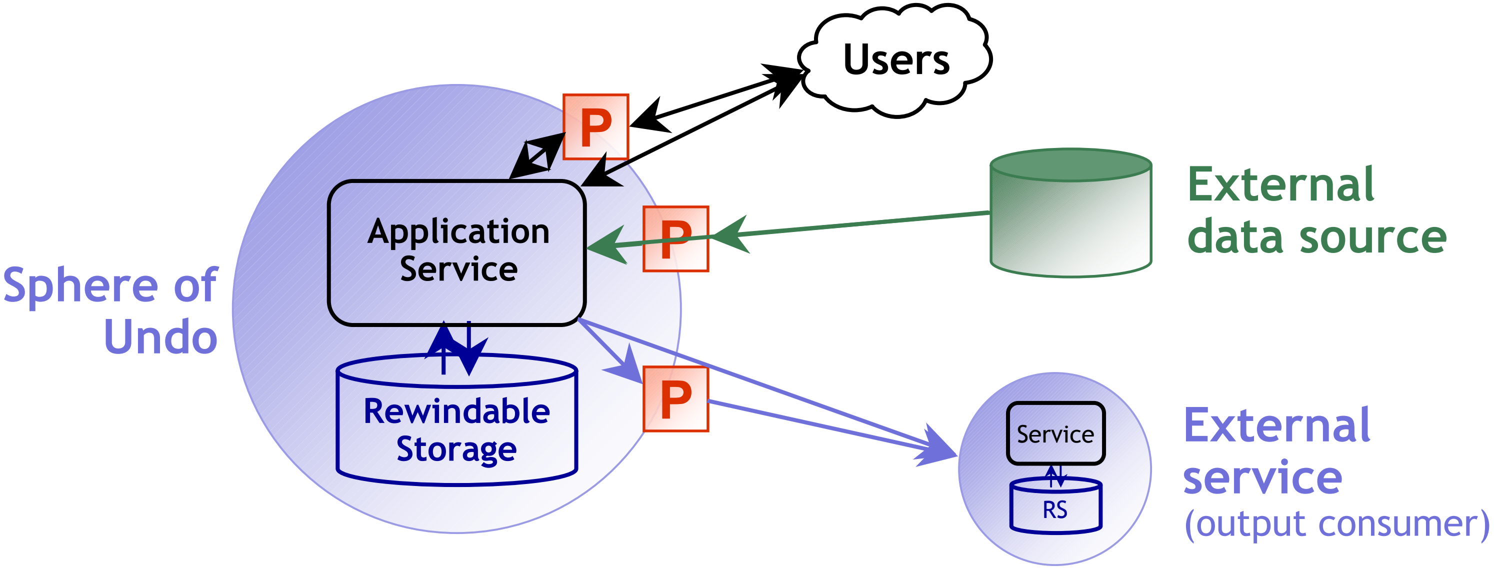
file/block storage service

missile launch control

- **When is undo possible?**
  - – state is centralized (or observable)
  - – all output to external entities can be intercepted
    - » and can be correlated to user requests
  - – external output is <u>provisional</u> for some time window
    - » e.g., can be cancelled, altered, reissued
    - » or simply doesn't matter in application's external consistency model

Extending Undo: Spheres of Undo

- Rewindable storage defines a sphere of undo



- All info crossing sphere must be intercepted
  - **input:** becomes verbs
  - **output:** becomes externalized output
    - » must be possible to associate output with a verb

Further Extensions

- **Verb concept may have broader applicability**
  - impact analysis of configuration changes
    - » use verb log as annotated history to evaluate changes on cloned system
  - self-checking data set for self-testing components
  - general approach to defining & encapsulating application consistency from end-user point of view?
    - » today, procedural and implicit
    - » can verbs be made declarative?
    - » can verbs be extracted automatically from object relationships?

More Verb Extensions

- **Extending verbs to administrative tasks**
  - in desktop environment
    - manage software installations/upgrades
    - provide “system refresh” using undo techniques
    - capture configuration changes at intent level
  - in server environment
    - move common tasks into undo framework
    - dynamically identify and guide ongoing operations tasks by analyzing verb sequences
  - key challenge in either environment is to capture breadth of administrative tasks

Conclusions

- **E-mail implementation demonstrates feasibility of Undo**
  - improvements in protocols, base storage technology would help reduce overhead
- **Human experiments to evaluate usefulness about to begin**
- **Verb construct has significant potential for further research**
  - extending Undo to broader domains
  - exploring other tools to support human operators

Undo: Update and Futures

- **Acknowledgements**
  - ROC Undergraduate Benchmarking Group
    - » Leonard Chung, Billy Kakes, Calvin Ling
  - Berkeley/Stanford ROC Research Group
- **For more info:**
  - abrown@cs.berkeley.edu
  - http://roc.cs.berkeley.edu/