Undo for Recovery: Approaches and Models

Aaron Brown

UC Berkeley ROC Group

ROC Retreat, January 2002

Outline

- Motivation
- The Three R's: An Undo paradigm
- Undo challenges
- System models for Undo
- Status

Motivation

- **Human operator error is a major cause of system failures**
  - people make mistakes, especially under stress
  - systems are not tolerant of human error during system administration
- **Undo is an effective paradigm for tolerating human error**
  - *recovery*-based: repairs unanticipated mistakes
  - familiar model: ubiquitous in productivity applications
    - » but rarely found in sysadmin tools

Motivation (2)

- **Undo "fringe benefits"**
  - makes sysadmin's job easier, improving maintainability
    - » better maintainability => better dependability
  - enables trial-and-error learning
    - » builds sysadmin's understanding of system
  - helps shift recovery burden from sysadmin to users
    - » export recovery to users via familiar undo model
    - » example: NetApp snapshots for file restores
  - helps recover from more than just human error
    - » SW/HW failure, security breaches, virus infections, ...

Outline

- Motivation
- **The Three R's: An Undo paradigm**
- Undo challenges
- System models for Undo
- Status

An Undo paradigm

- **ROC Undo combines time travel with repair**
- **The Three R's of Undo**
  - **Rewind**: roll system state backwards in time
  - **Repair**: fix latent or active errors
    - » automatically or via human intervention
  - **Replay**: roll system state forward, replaying user interactions lost during rewind
    - » we assume a service model with well-defined user actions
- **All three steps are critical**
  - rewind enables undo
  - repair lets user/administrator fix problems
  - replay preserves updates, propagates fixes forward

Undo examples: email

- **Coarse-grained Undo**
  - roll back OS or app. upgrade without losing user data
  - revert system-wide configuration changes
  - "go back in time" to retroactively install virus filter on email server; effects of virus are squashed on redo
- **Fine-grained Undo**
  - undo deletion of a user, mailbox, or email message
  - reverse changes to a user's profile or filtering rules
  - maybe even unsend mail (?)
- **Undo paradigm must support both granularities**

Context

- **Traditional Undo gives only two R's**
  - undo/repair or undo/replay
  - includes models presented at last retreat
- **What about checkpoints, reboot, DB logging?**
  - checkpoints gives Rewind only
  - reboot may give Repair, but only for "Heisenbugs"
  - logging can give all 3 R's
    - » but RDBMS logging is insufficient, since system state changes are interdependent and non-transactional
    - » 3R-logging requires understanding of dependencies and attention to externalized events

Outline

- Motivation
- The Three R's: An Undo paradigm
- **Undo challenges**
- System models for Undo
- Status

Challenges in 3R paradigm

- **Integrated coarse- and fine-grained undo**
  - coarse-grained undo best handled by physical logging
    - » disk snapshots, non-overwriting disks, etc.
    - » operates below the OS/application layer
  - fine-grained undo best handled by logical logging
    - » at application level, where event semantics are known
  - best: hybrid system with physical logging for Rewind and logical logging for Replay
    - » caveat: limits full 3R semantics to logically-logged system state; allows simple undo/redo of unmonitored state
      - i.e., redo of unmonitored state won't propagate repairs

Challenges in 3R paradigm (2)

- **Handling externalized events**
  - externalized event: event visible outside of system
    - » example: user downloads/reads email message
    - » example: user forwards email message over the Internet
  - undo can invalidate externalized events
    - » repair can cause events to change/disappear on replay
    - » result: inconsistency between system and external env't
  - solutions depend on acceptable level of inconsistency
    - » human users willing to accept inconsistency in some apps
    - » issue compensating or explanatory events
    - » delay execution of externalized events for an undo window
    - » convert external to internal by expanding system boundary

Challenges in 3R paradigm (3)

- **Managing state dependencies**
  - Rewind/Repair cycle can invalidate logged events
    - » example: retroactive installation of virus filter invalidates delivery of virus-laden messages
  - Replay system must understand dependencies between logged state and state touched during repair
- **Managing state logs**
  - what events and user interactions to log for redo?
    - » include administrative events like SW installs?
  - how to present logs to human for editing during repair
- **Minimizing performance and overhead penalties**
  - for now, we assume abundant disk and CPU resources

Outline

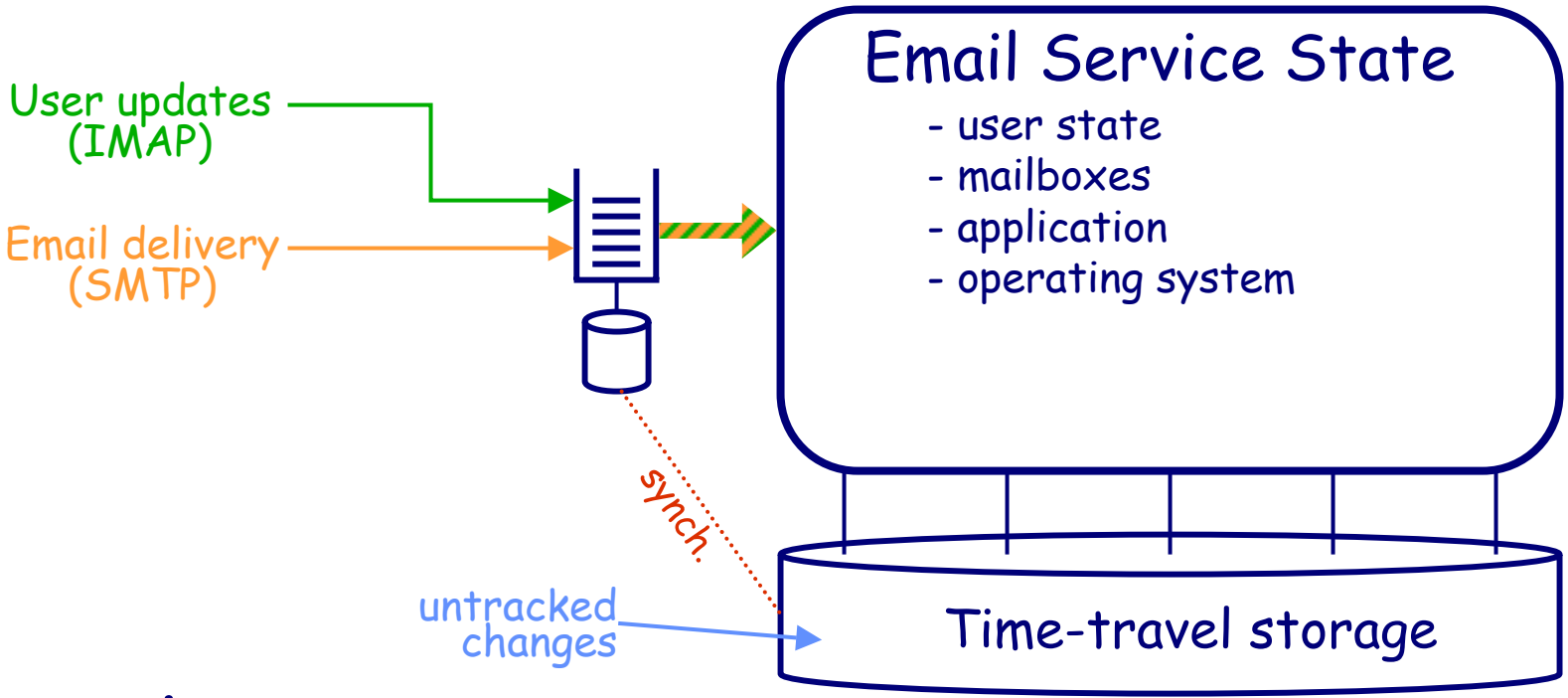
- Motivation
- The Three R's: An Undo paradigm
- Undo challenges
- **System models for Undo**
- Status

Towards system models for undo

- **Goal: abstract model for undo-capable system**
  - template for constructing undoable services
  - needed to analyze generality and limitations of undo
- **Model components**
  - state entities
  - state update events (analogue of transactions)
  - event queues and logs
  - untracked system changes
- **Assumptions**
  - storage layer that supports bidirectional time-travel
    - » via non-overwriting FS, snapshots, etc.
- **Email as example application**

Simple model

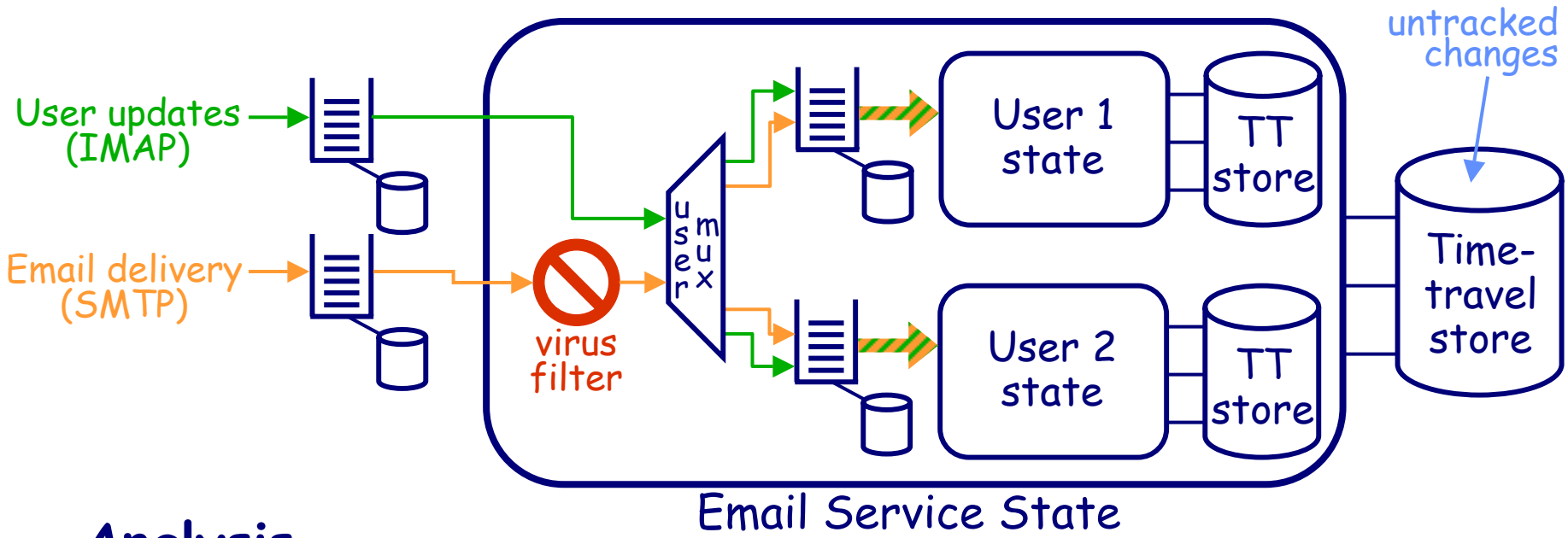
- **Entire system is one state entity**



- **Analysis**
  - + simple, easy to implement, easier to trust, most general
  - – huge overhead for fine-grained undo operations
  - – serialization bottleneck at single queue/log
  - – difficult to distinguish different users' events

Hierarchical model

- **System composed of multiple state entities**
  - each state entity supports undo as in simple model
  - state entities join hierarchically to give multiple granularities of undo



- **Analysis**
  - + multiple undo granularities reduces overhead, bottlenecks
  - + distributed undo possible
  - – greater complexity; tricky to coordinate different layers

Outline

- Motivation
- The Three R's: An Undo paradigm
- Undo challenges
- System models for Undo
- **Status**

Status

- **Refining system model for undo**
  - hierarchical seems best bet, but many issues to solve
  - feedback welcome!
- **Learning about real-world email systems**
  - to help calibrate the undo model
  - working with Sun/iPlanet Messaging Server team
    - » likely will get source code access
- **Continuing maintainability benchmarking work**
  - helps illustrate what kinds of things need undo
- **Preparing for proof-of-concept implementation**
  - in the context of iPlanet Messaging Server