Addressing Human Error with Undo

Aaron Brown

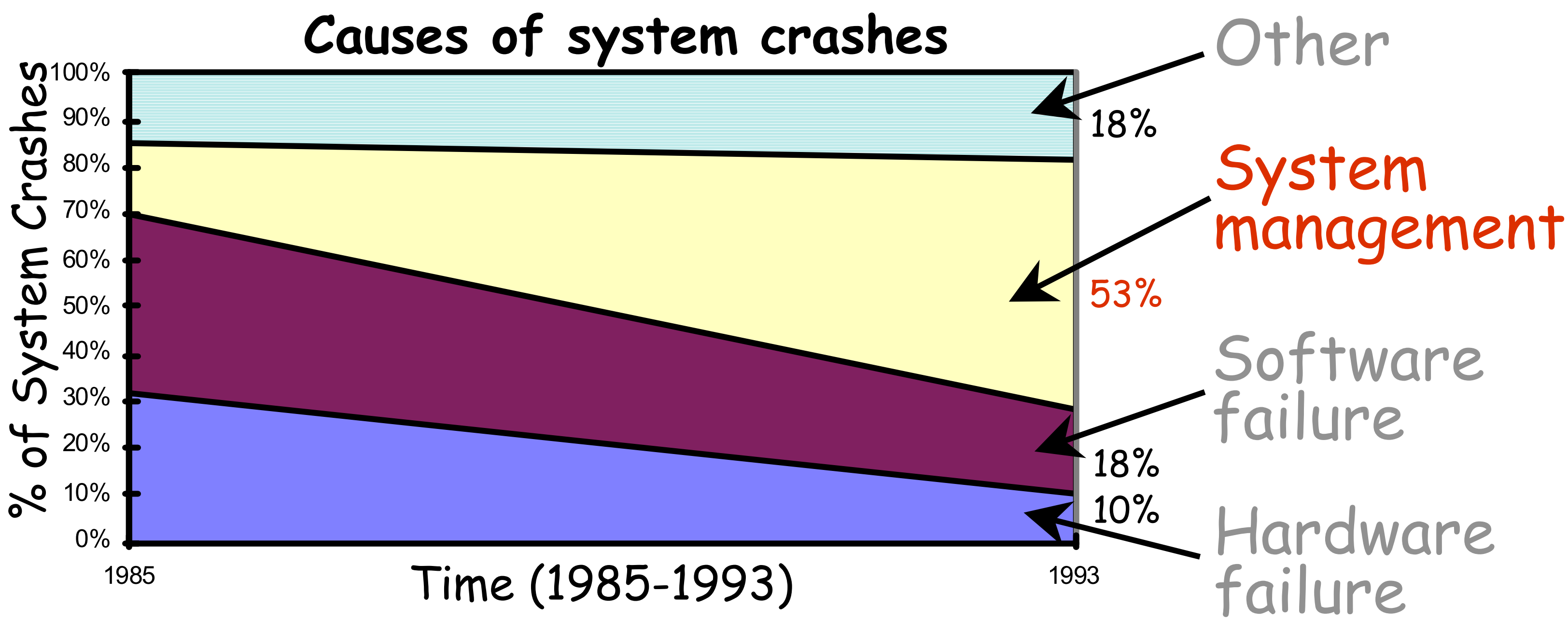
ROC Retreat, June 2001

Outline

- **Motivation:** importance of human error during system maintenance
- **Challenge:** providing recovery from human error
- **Solution:** undo
  - defining an undo paradigm for system administration
  - implementation techniques for sysadmin undo
- **Status and plans**

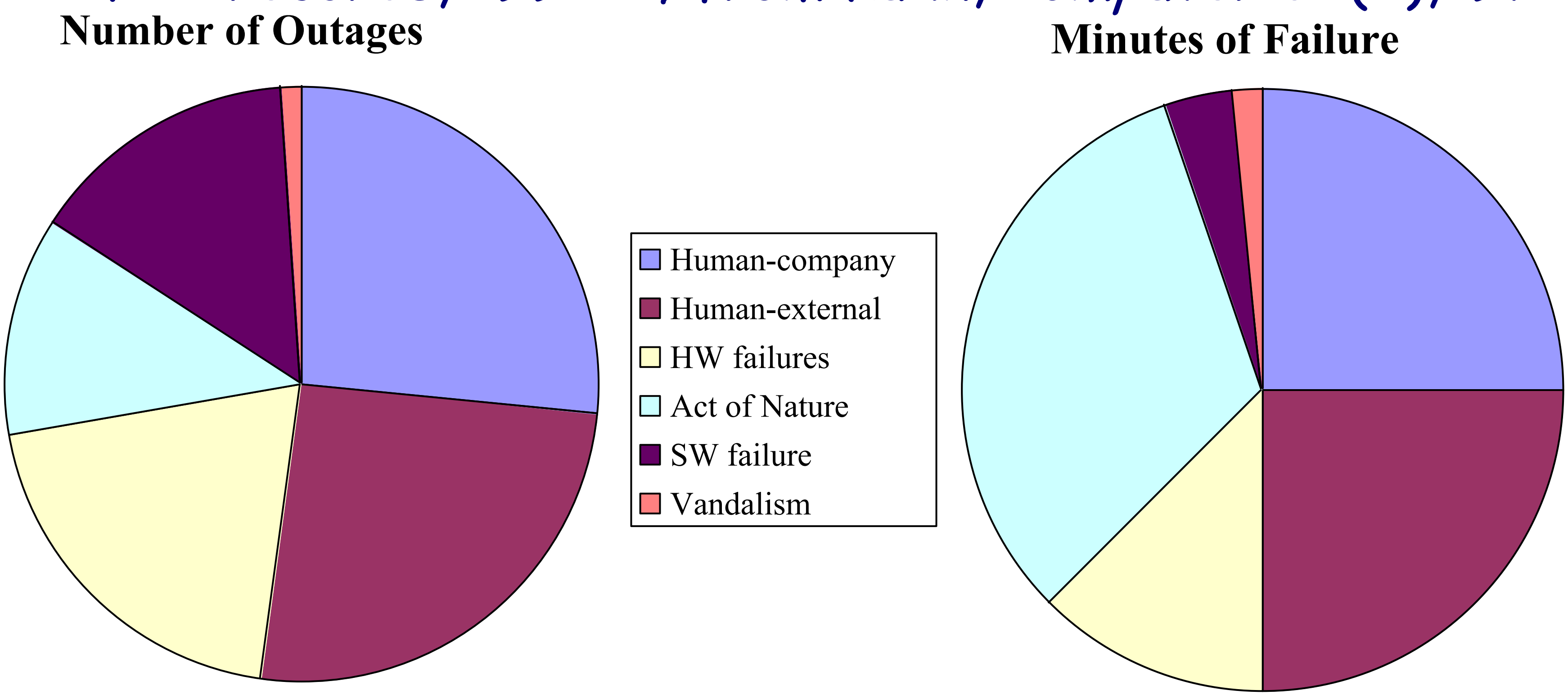
Motivation: human error is important

- **Half of system failures are from human error**
  - Oracle: half of DB failures due to human error (1999)
  - Gray/Tandem: 42% of failures from human administrator errors (1986)
  - Murphy/Gent study of VAX systems (1993):



Human error is important (2)

- **More data: telephone network failures**
  - FCC records, 1992-4; from Kuhn, *Computer* 30(4), '97



  - half of outages, outage-minutes are human-related
    - » about 25% are direct result of maintenance errors by phone company workers

Don't just blame the operator!

- **Psychology shows that human errors are inevitable** [see J. Reason, *Human Error*, 1990]
  - humans prone to *slips* & *lapses* even on familiar tasks
    - » 60% of errors are on "skill-based" automatic tasks
  - also prone to *mistakes* when tasks become difficult
    - » 30% of errors on "rule-based" reasoning tasks
    - » 10% of errors on "knowledge-based" tasks that require novel reasoning from first principles
- **Allowing human error can even be beneficial**
  - mistakes are a part of trial-and-error reasoning
    - » trial & error is needed to solve knowledge-based tasks
    - » fear of error can stymie innovation and learning

Outline

- **Motivation**: importance of human error during system maintenance
- **Challenge**: providing recovery from human error
- **Solution**: undo
  - defining an undo paradigm for system administration
  - implementation techniques for sysadmin undo
- **Status and plans**

Recovery from human error

- **ROC principle: recovery from human error, not avoidance**
  - accepts inevitability of errors
  - promotes better human-system interaction by enabling trial-and-error
    - » improves other forms of system recovery
- **Recovery mechanism: Undo**
  - ubiquitous and well-proven in productivity applications
  - unusual in system maintenance
    - » primitive versions exist (backup, standby machines, ...)
    - » but not well-matched to human error or interaction patterns

Outline

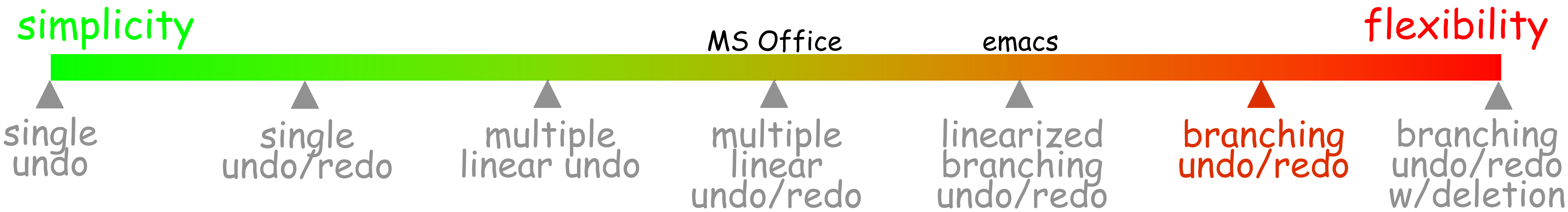
- **Motivation**: importance of human error during system maintenance
- **Challenge**: providing recovery from human error
- **Solution**: undo
  - defining an undo paradigm for system administration
  - implementation techniques for sysadmin undo
- **Status and plans**

Undo paradigms

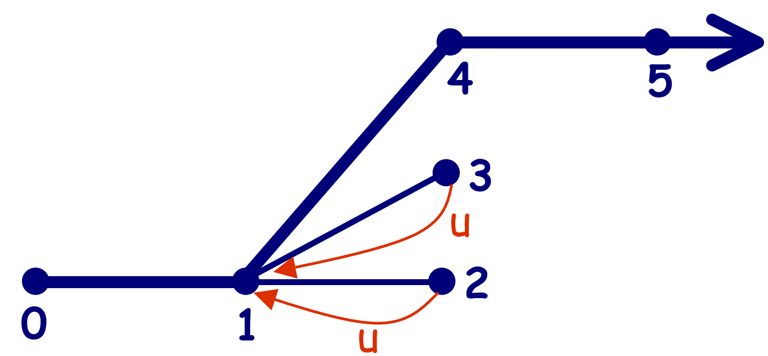
- **An effective undo paradigm matches the needs of its target environment**
  - cannot reuse existing undo paradigms for system maintenance
- **We need a new undo paradigm for maintenance**
  - plan:
    - » lay out the design space
    - » pick a tentative undo paradigm
    - » carry out experiments to validate the paradigm
- **Underlying assumption: service model**
  - single application
  - users access via well-defined network requests

Issue #1: Choice of undo model

- Undo model defines the view of past history
- Spectrum of model options:



- Important choices:
  - undo only, or undo/redo?
  - single, linear, or branching?
  - deletion or no deletion?
- Tentative choice for maintenance undo



trial-and-error history pattern

More undo issues

2) Representation

- does undo act on **states** or **actions**?
- how are the states/actions named? TBD

3) Selection of undo points

- granularity:
  - » undo points at each state change/action?
  - » or at checkpoints of some granularity?
- are undo points administrator- or system-defined?

- **Tentative maintenance undo choices in red**

More undo issues (2)

4) Scope of undo

- "what state can be recovered by undo?"
- single-node, multi-node, multi-node+network?
- on each node:
  - » system hardware state: BIOS, hardware configs?
  - » disk state: user, application, OS/system?
  - » soft state: process, OS, full-machine checkpoints?
- tentative maintenance undo goals in red

More undo issues (3)

5) Transparency to service user

- ideally:
  - » undo of system state preserves user data & updates
  - » user always sees consistent, forward-moving timeline
  - » undo has no user-visible impact on data or service availability

Context: other undo mechanisms

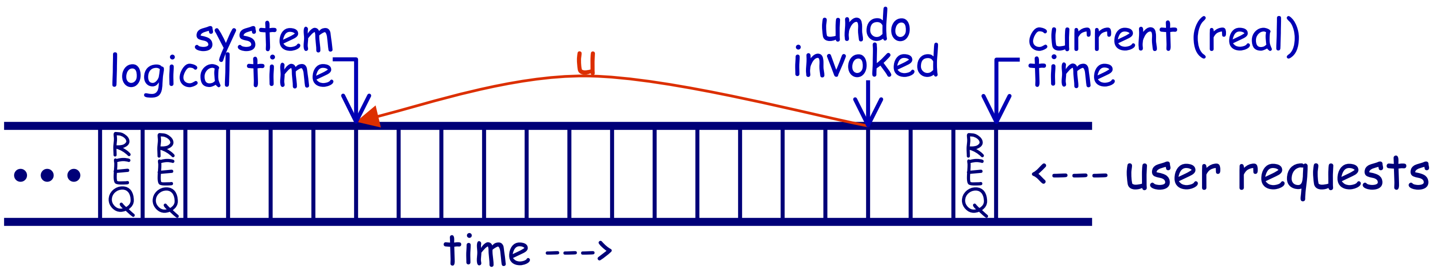
Design axis / Undo mech.	Undo model	Representation	Undo-point selection	Scope	Trans-parency
Desired maintenance-undo semantics	branching undo/redo	state, naming TBD	automatic checkpoints	all disk & HW, all nodes & network	high
Geoplex site failover	single undo	state, unnamed	varies; usu. automatic checkpoints	entire system	high
Tape backup	single or multiple linear undo	state ad-hoc naming	manual checkpoints	disk (1 FS), single node	low
GoBack®	linearized branching undo/redo	state, temporal naming	automatic checkpoints	disk (all), single node	low-medium
Netapp Snapshots	multiple linear undo	state, temporal naming	manual checkpoints	disk (all), single server	low
DBMS logging (for txn abort)	single undo	hybrid, unnamed	automatic checkpoints	single txn, app-level	high

Implementing maintenance undo

- **Saving state: disk**
  - apply snapshot or logging techniques to disk state
    - » e.g., NetApp- or VMware-style block snapshots, or LFS
    - » all state, including OS, application binaries, config files
  - leverage excess of cheap, fast storage
  - integrate "time travel" with native storage mechanism for efficiency
- **Saving state: hardware**
  - periodically discover and log hardware configuration
  - can't automatically undo all hardware changes, but can direct administrator to restore configuration

Implementing maintenance undo (2)

- **Providing transparency**
  - queue & log user requests at edge of system, in format of original request protocol
  - correlate undo points to points in request log
  - snoop/replay log to satisfy user requests during undo



- **An undo UI**
  - should visually display branching structure
  - must provide way to name and select undo points, show changes between points

Outline

- **Motivation:** importance of human error during system maintenance
- **Challenge:** providing recovery from human error
- **Solution:** undo
  - defining an undo paradigm for system administration
  - implementation techniques for sysadmin undo
- **Status and plans**

Status and plans

- **Status**
  - starting human experiments to pin down undo paradigm
    - » subjects are asked to configure and upgrade a 3-tier e-commerce system using HOWTO-style documentation
    - » we monitor their mistakes and identify where and how undo would be useful
  - experiments also used to evaluate existing undo mechanisms like those in GoBack and VMware
- **Plans**
  - finalize choice of undo paradigm
  - build proof-of-concept implementation in Internet email service on ROC-1 cluster
  - evaluate effectiveness and transparency with further experiments