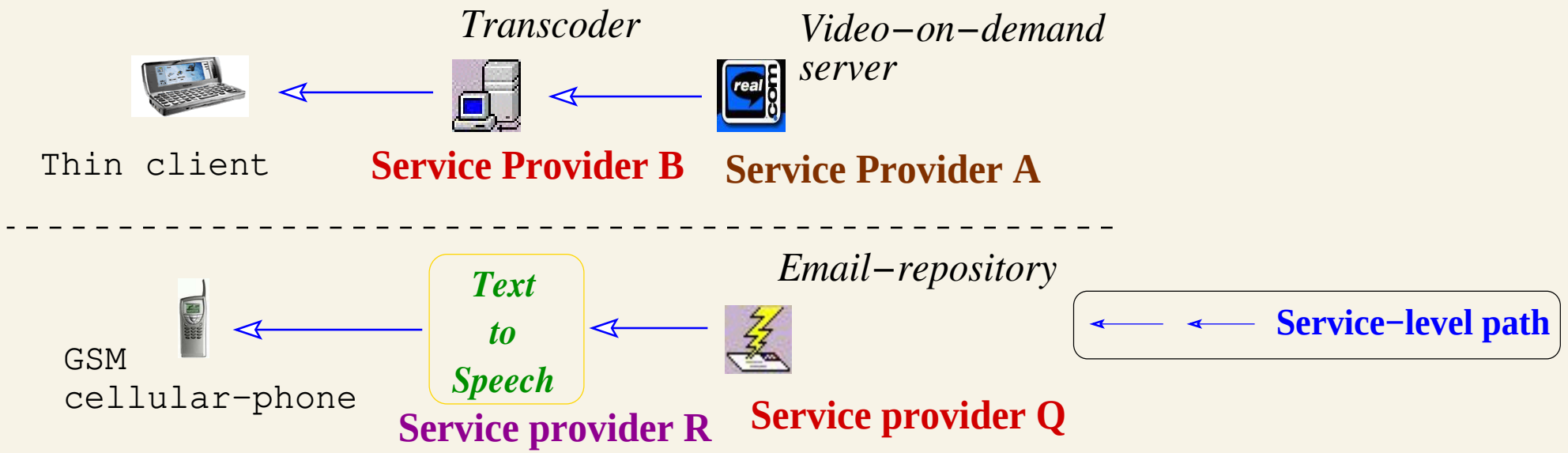
An Architecture for Highly Available Wide-Area Service Composition

Bhaskaran Raman
Dissertation Talk
EECS, U. C. Berkeley

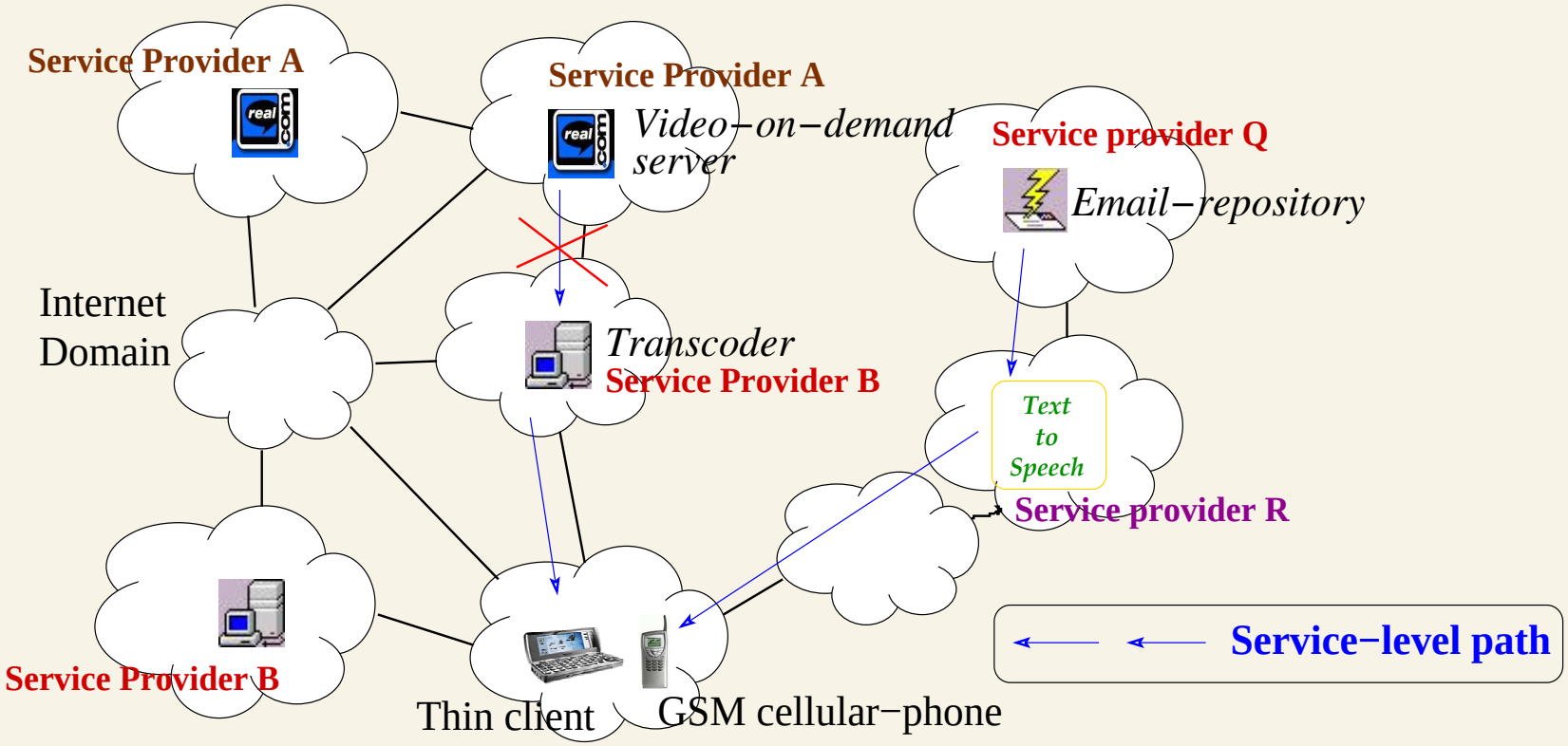
05 Dec 2002

Service Composition

- Composition: assembling independent components
- Example: Unix piping
  - gunzip --stdout file.gz | grep string1 | awk '{print $2;}'
- Quickly enable new functionality, flexibility
- Wide-Area Service Composition
  - Assemble service components from independent service providers
- Examples:



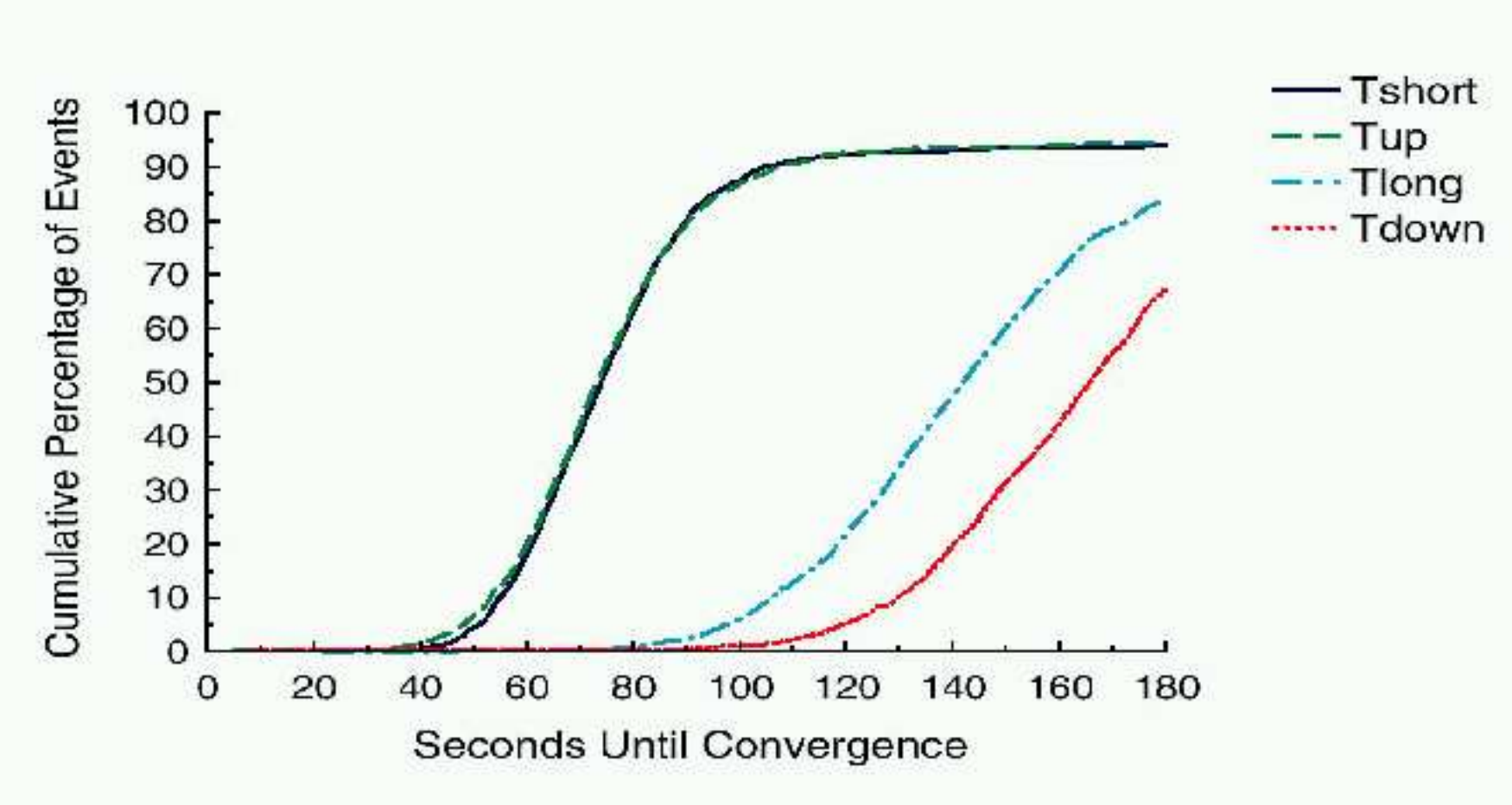
Issue of Availability



- Service-level path could span the wide-area Internet
- BGP: Inter-domain Internet routing protocol
- Can fail frequently
  - Availability can be as low as 95% [Labovitz et al, FTCS'99]

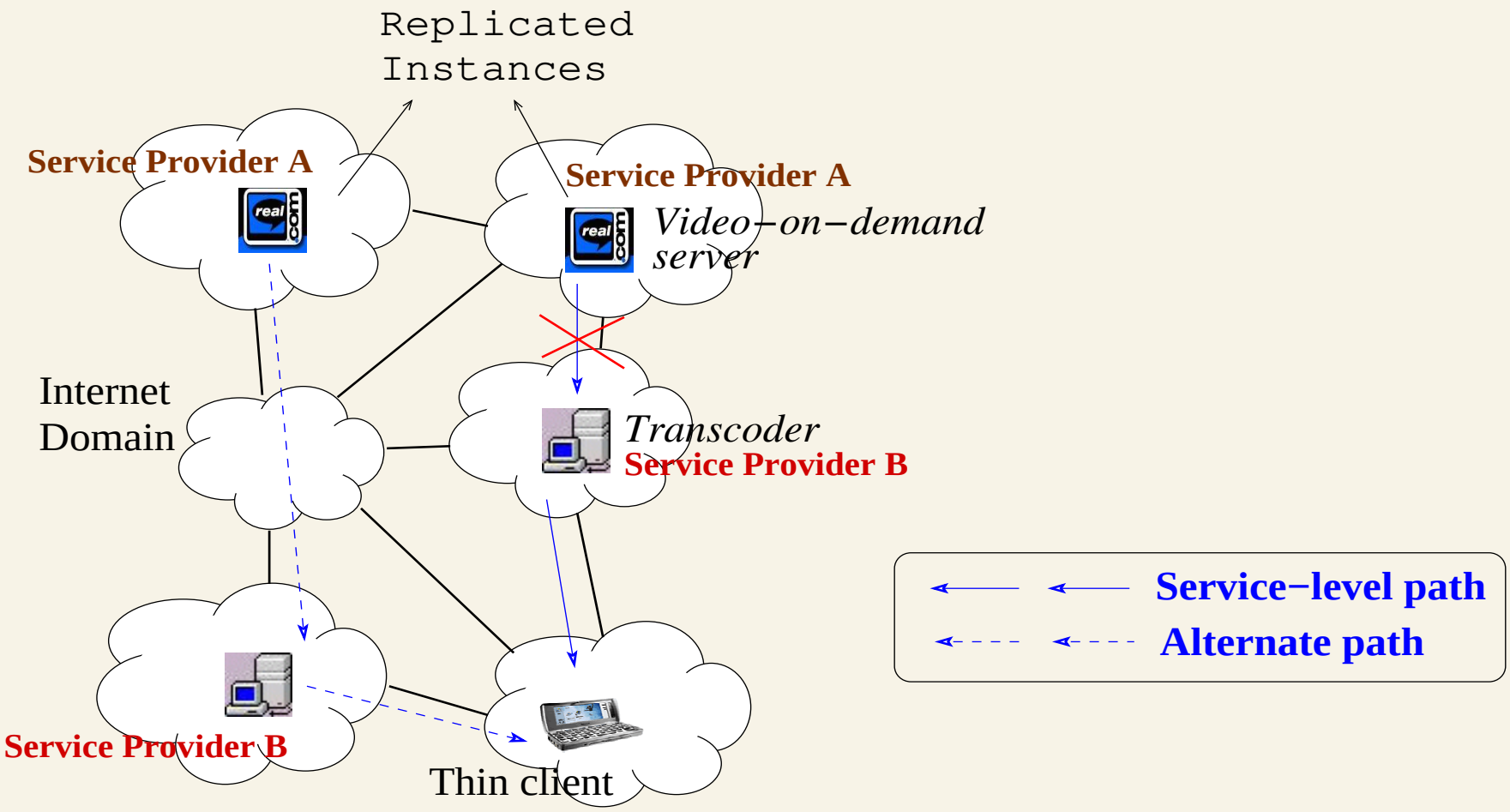
Issue of Availability (continued)

- BGP can take up to several minutes to recover [Labovitz et al, SIGCOMM'00]



Solution Approach

- Use service replicas, and alternate paths



Goals

- Availability: Detect and fix failures **quickly**
- Performance, Load Balancing: choose least loaded service instances

- Scalability: Internet-scale operation, scale with number of clients

Outline

- Motivation and goals
- Related work
- Research issues and Challenges

- Failure detection in the wide-area Internet
- Architecture for wide-area composition
- Evaluation

- Summary of contributions, future directions

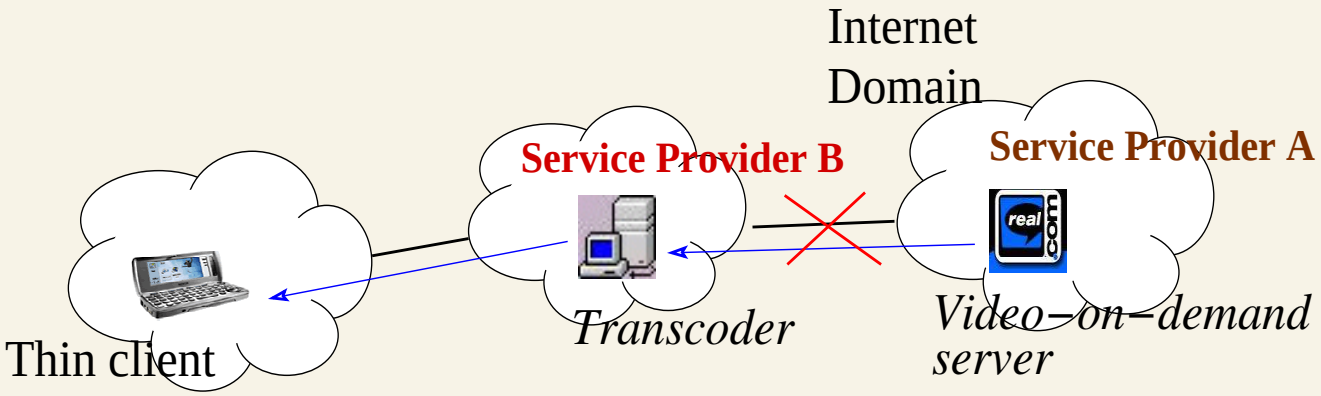
Related Work

- Service composition
  - Fault tolerance: TACC (Berkeley)
    - Within a single cluster, using intra-cluster monitoring
  - Service interfacing: Web-Services, IETF Open Pluggable Edge Services
    - Web-Services Description/Flow Language, Intermediary Rule Markup Language
- Availability and performance of web-servers: SPAND (Berkeley), Harvest (Colorado)
  - Not for composed services
  - Fault-tolerance during a session not considered
- Using alternate Internet paths: overlay networks
  - CAN (Berkeley), Chord (MIT), Tapestry (Berkeley), RON (MIT)
  - Have not considered service composition

Availability issues in wide-area service composition not addressed so far

Research issues and Challenges

- Failure detection



  - What is a failure on a wide-area Internet path?
    - Is there a clearly defined notion of "failure"?
    - Can it be distinguished from "intermittent congestion" or "short outages"?
    - What mechanism to use to detect?
    - How quickly can it be detected?
- Scaling
  - Is an issue since global information needed
  - Information about location of all services, all network paths in-between

Outline

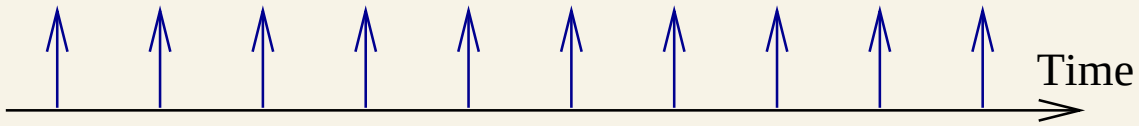
- Motivation and goals
- Related work
- Research issues and Challenges

- Failure detection in the wide-area Internet
- Architecture for wide-area composition
- Evaluation

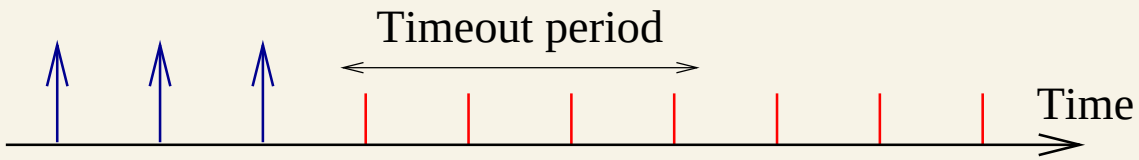
- Summary of contributions, future directions

Failure Detection: What and How

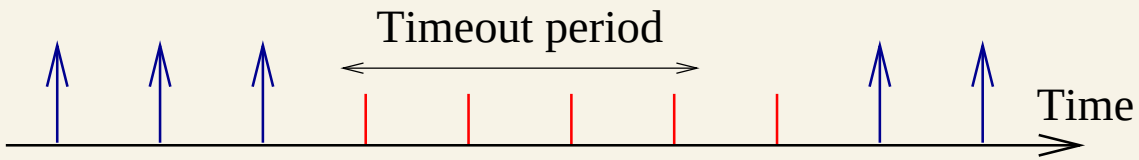
- What is a failure in an Internet path?
  - When application is no more useful to user
    - Very application dependent notion
  - For this work, we consider outages in connectivity
- How to detect such outages?



(a) Monitoring for liveness of path using keep−alive heartbeat



(b) Failure: detected by timeout

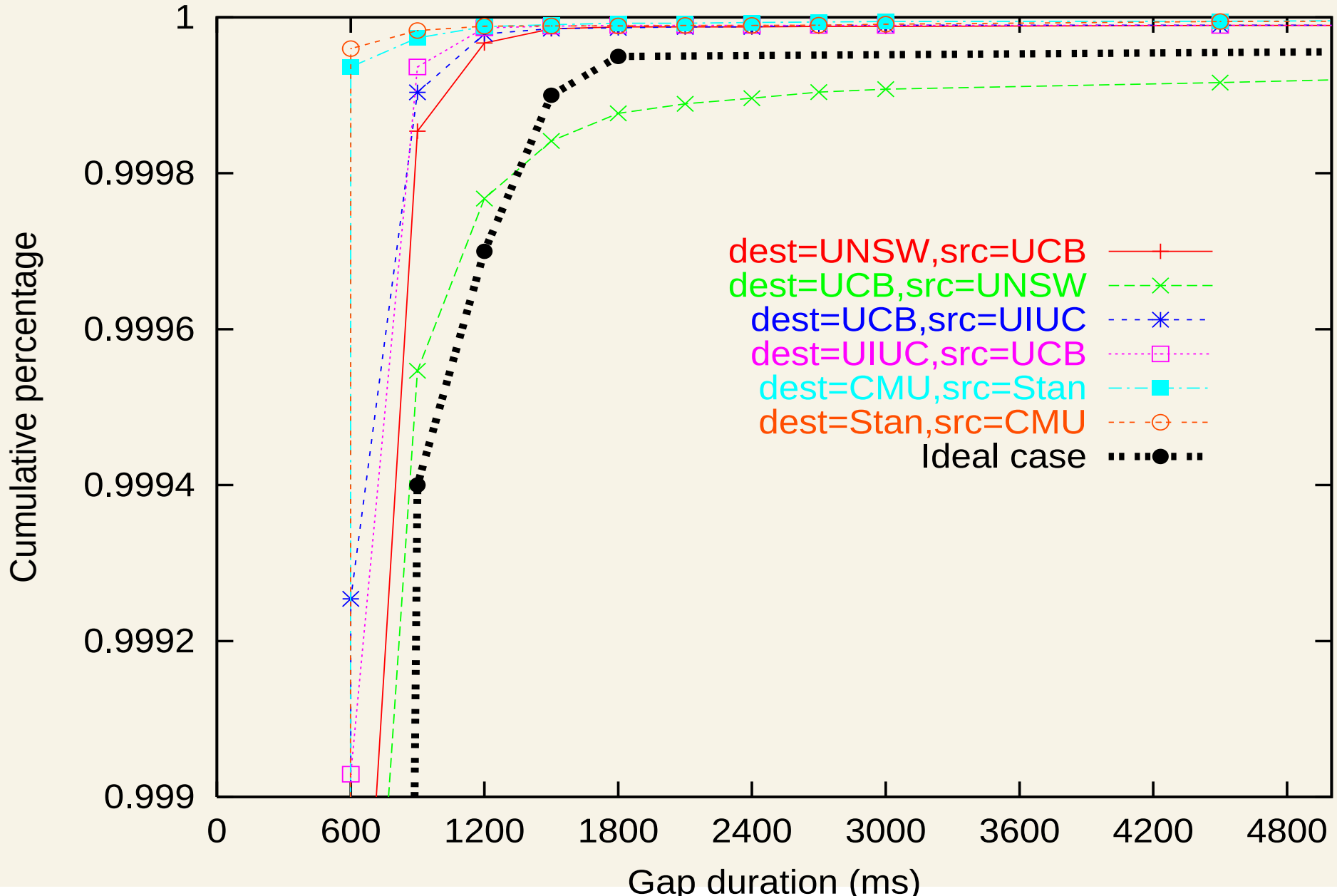


(c) False−positive: failure detected incorrectly due to intermittent congestion

Failure Detection Mechanism

- Questions to answer:
  - What is the heart-beat period?
  - What should be the failure detection timeout?
  - How often is intermittent congestion or short outage confused with a failure?
- Need to know distribution and probability of gaps
- Approach: trace collection over wide-area Internet paths
  - 6 different Internet hosts
    - Berkeley, Stanford, UIUC, CMU, TU-Berlin, UNSW
  - 18 different Internet paths between these
    - End-to-end latency varies between 3 ms to 220 ms
    - #Internet domains in path varies from 3 to 6
  - 300 ms heart-beat along Internet path
  - Collect data over 4-7 days
  - Plot distribution of gaps seen at receiving end

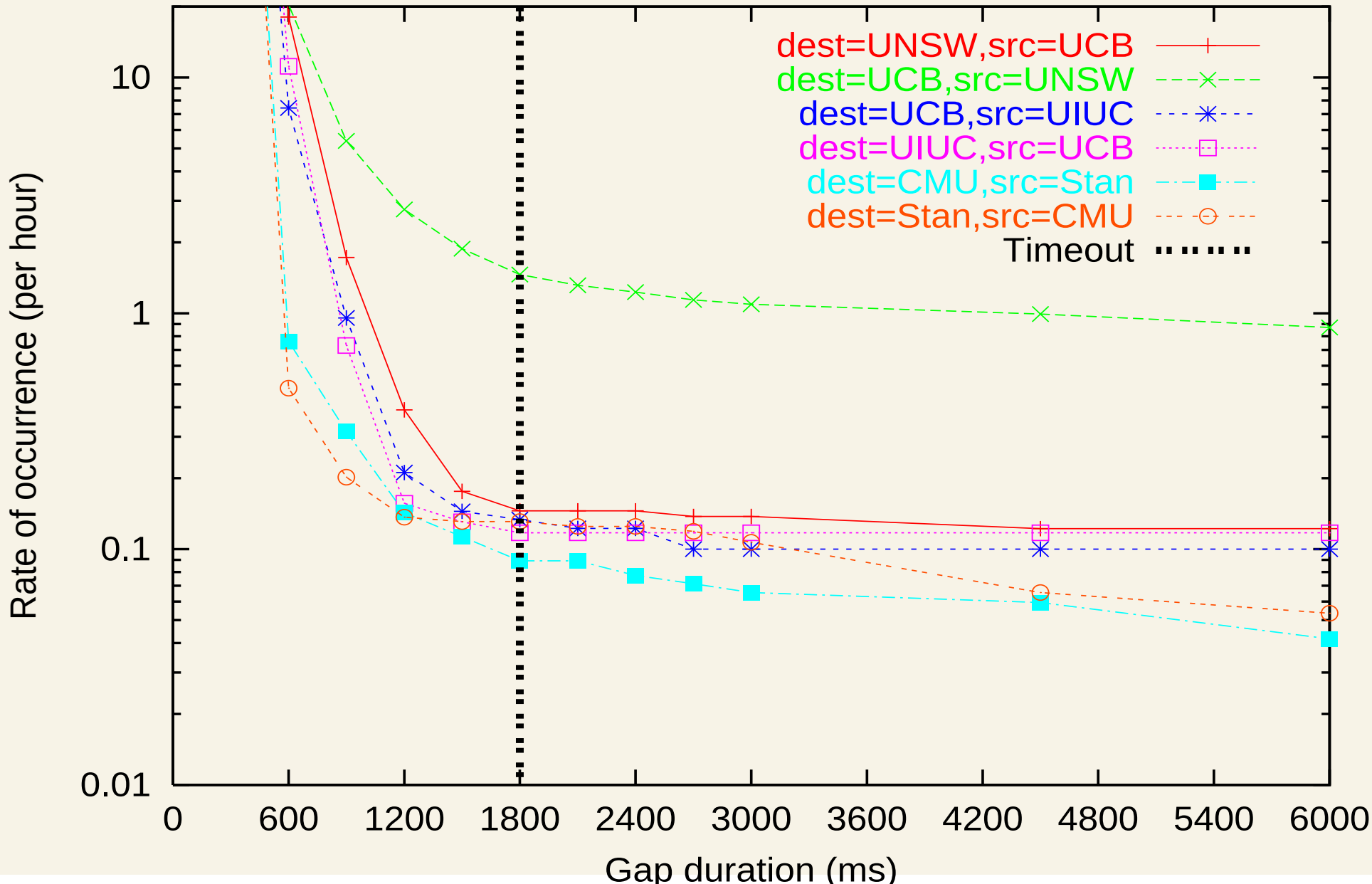
CDF of gap durations



Implications of the CDF

- Reality "close" to ideal
  - Ideal case: Timeout (1.8sec) ==> Long outage (30sec) with probability 1
  - In reality: this probability is less than 1
    - Over 40% for 4 of the 18 Internet paths
    - 15-40% for another 6
  - Is this good enough?
    - Prediction probability is 15% ==> 85% false-positives
    - Extra work done to handle false-positives
  - Two questions to answer:
    - What's the cost of a timeout?
    - How often do timeouts occur in absolute terms?

Gaps: rate of occurrence



Failure Detection: Summary

- Timeout of about 1.5-2 seconds can be used to detect outages in wide-area Internet paths
  - Predicts long outages with "good" probability
  - Low rate of occurrence of timeouts
  - May have to be determined adaptively for an Internet path
- If failure recovery can be done in about 1-2 seconds after failure detection
  - Would be much better than relying upon Internet path recovery
    - Several tens of seconds to a few minutes
  - Would be immensely useful for real-time applications

Outline

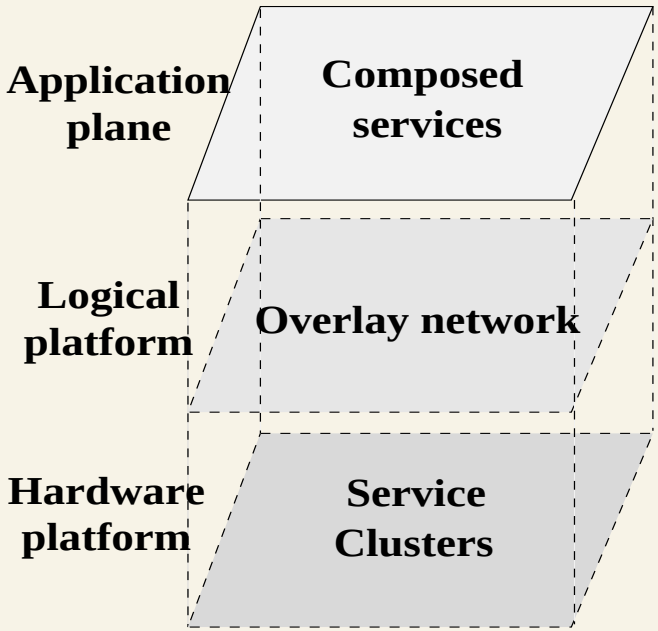
- Motivation and goals
- Related work
- Research issues and Challenges

- Failure detection in the wide-area Internet
- Architecture for wide-area composition
- Evaluation

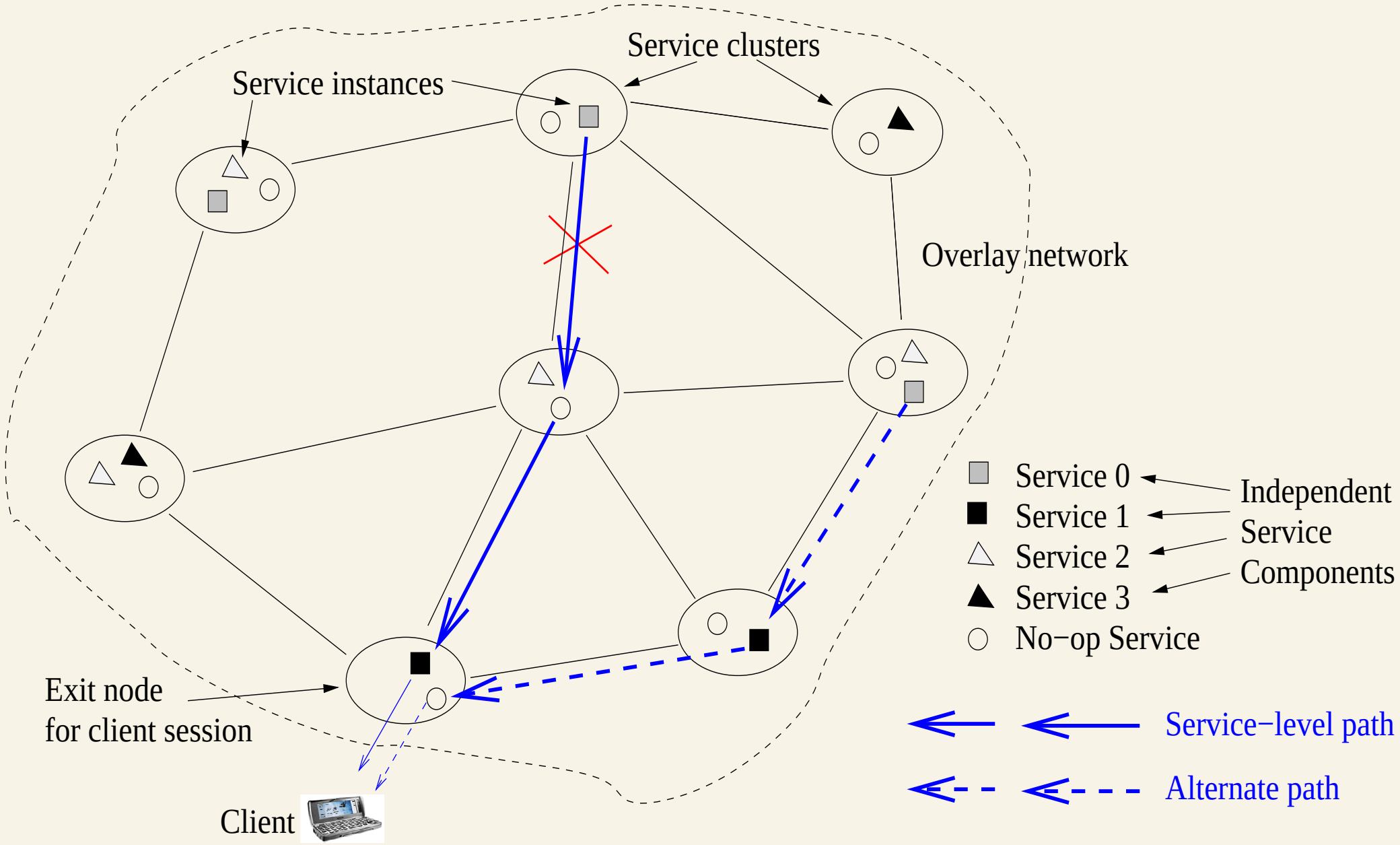
- Summary of contributions, future directions

Towards an Architecture

- Service execution platforms for providers to deploy services
  - Each execution platform is a service cluster
- Overlay network of such execution platforms
  - To enable composition (service-level paths)
  - To exploit redundancy in Internet paths



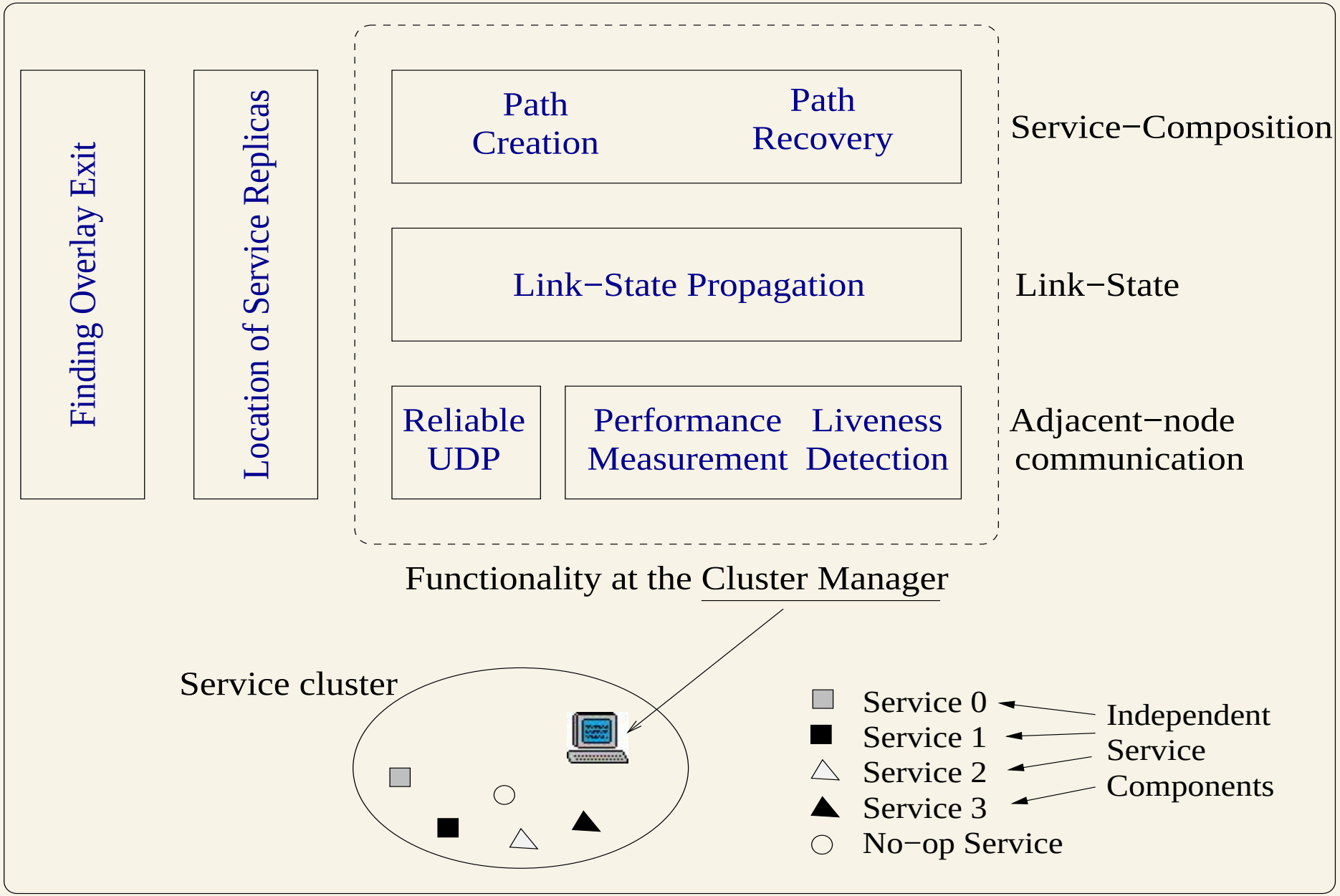
Architecture



Key Design Points

- Use of service clusters
  - Process/machine-level failures can be handled within cluster
  - Inter-cluster network path failure detection and other overheads amortized across clients
- What should be the overlay size?
  - Overlay should span the Internet
    - An exit-node "close" to every potential client
  - Number of Autonomous Systems: 12,000 (as of 2001)
  - Akamai: 1,000+ server locations (as of 2001)
    - Many more number of servers
  - A few thousand overlay nodes should be good enough
    - Can it be any lesser?

Software Architecture



Software Architecture (continued)

- Adjacent-node communication
  - Failure detection using heart-beats
- Link-state-based propagation
  - Quick propagation of failure information
  - Provides entire graph state
- Service composition
  - Algorithm for service-level path creation
    - Based on a modification to Dijkstra's algorithm
    - Setting metrics is an important issue: INFOCOM'03 paper
  - Service-level path recovery
    - Done by exit-node
    - Creates an alternate service-level path with current graph information

Outline

- Motivation and goals
- Related work
- Research issues and Challenges

- Failure detection in the wide-area Internet
- Architecture for wide-area composition
- Evaluation
  - Scaling overheads (overview)
  - Wide-area measurements using text-to-speech composed application

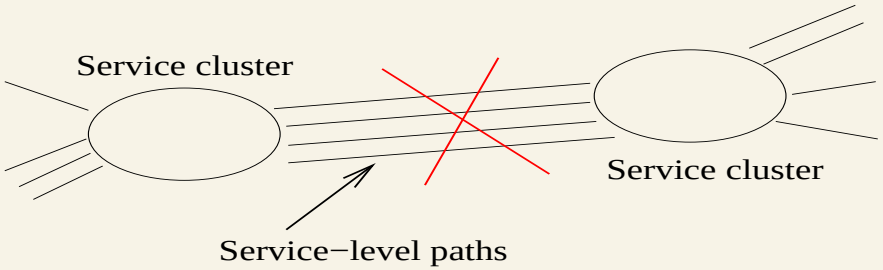
- Summary of contributions, future directions

Scaling and Link-State

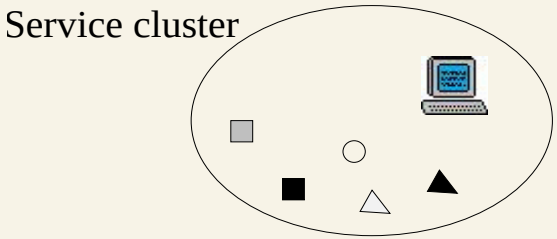
- Possible scaling bottlenecks
  - Network flooding, Memory, CPU computation
- Network flooding
  - For a 2,000-node graph, say with 10,000 edges
  - Failures, with a timeout of 1.8sec: about once an hour per edge
  - About 6 floods/sec in the entire network
    - Only 6 messages/sec on each edge due to all floods
- Memory, CPU
  - Service composition algorithm on 6,510-node network
  - Only a few MB of storage
  - Service-level path computation: 50ms
    - Path caching can be done
    - Incremental Dijkstra's can be applied

Scaling and Number of Clients

- Many client sessions may have to be recovered all at once



- Flurry of messages during recovery
  - Limit on num. of service-level paths on a graph edge ==>
  - Limit on num. of service-level paths per cluster-manager
  - At least 250-300 simultaneous paths per cluster-manager machine



  - In comparison, 10-15 simultaneous client sessions per machine, for the text-to-speech service
- More details in SPECTS'02 paper

Outline

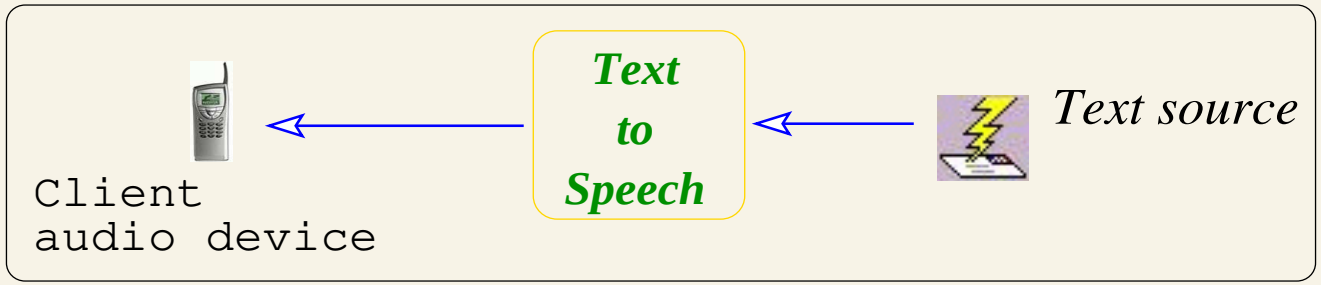
- Motivation and goals
- Related work
- Research issues and Challenges

- Failure detection in the wide-area Internet
- Architecture for wide-area composition
- Evaluation
  - Scaling overheads (overview)
  - Wide-area measurements using text-to-speech composed application

- Summary of contributions, future directions

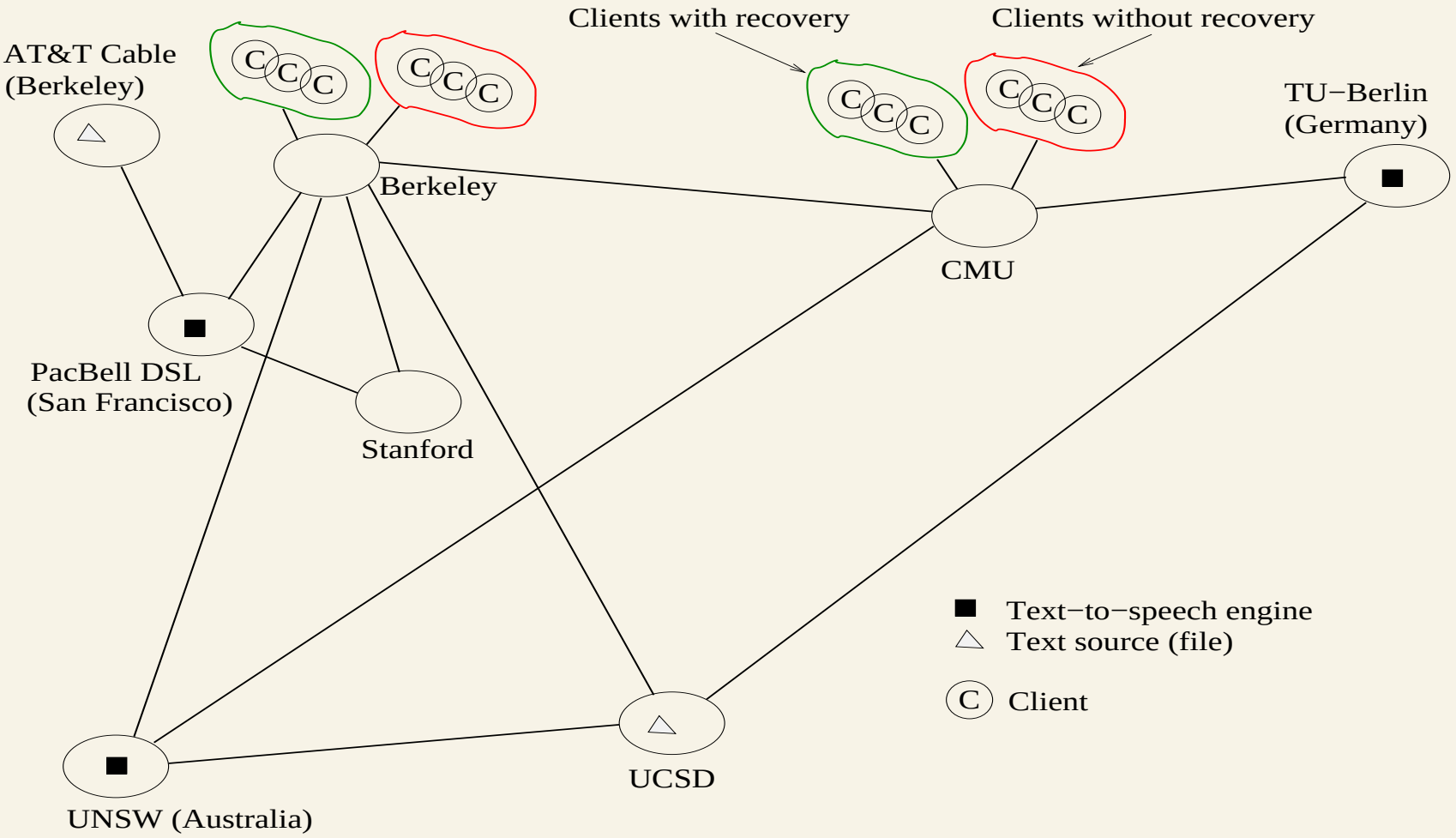
Wide-area experiments: setup

- Goal: test the usefulness of failure detection/recovery mechanisms
  - Using a real composed application
  - In a real wide-area testbed
- Composed application: text-to-speech service



Wide-area experiments: setup (continued)

- Wide-area testbed: 8 nodes



- Internet path latency: varied between 3 ms and 220 ms
- # AS domains in path: varied between 3 and 6

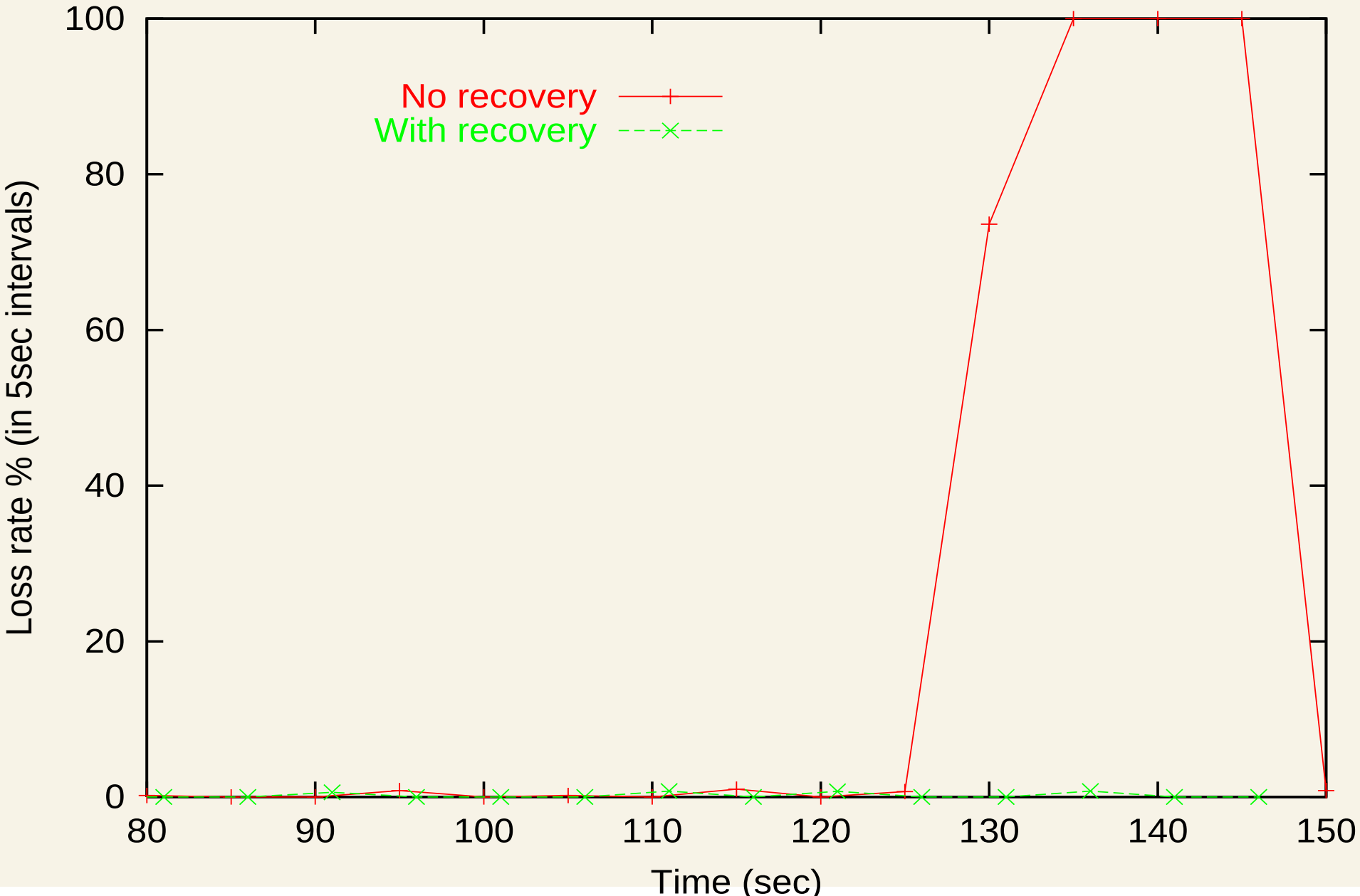
Wide-area experiments: setup (continued)

- Four paths in the system at any point of time
  - Each path with recovery enabled, has a counterpart with recovery disabled
- Duration of each session: 2min 30sec
- Run for 1 day at a time, for 11 days (gaps in-between)
  - Includes week-days as well as week-ends

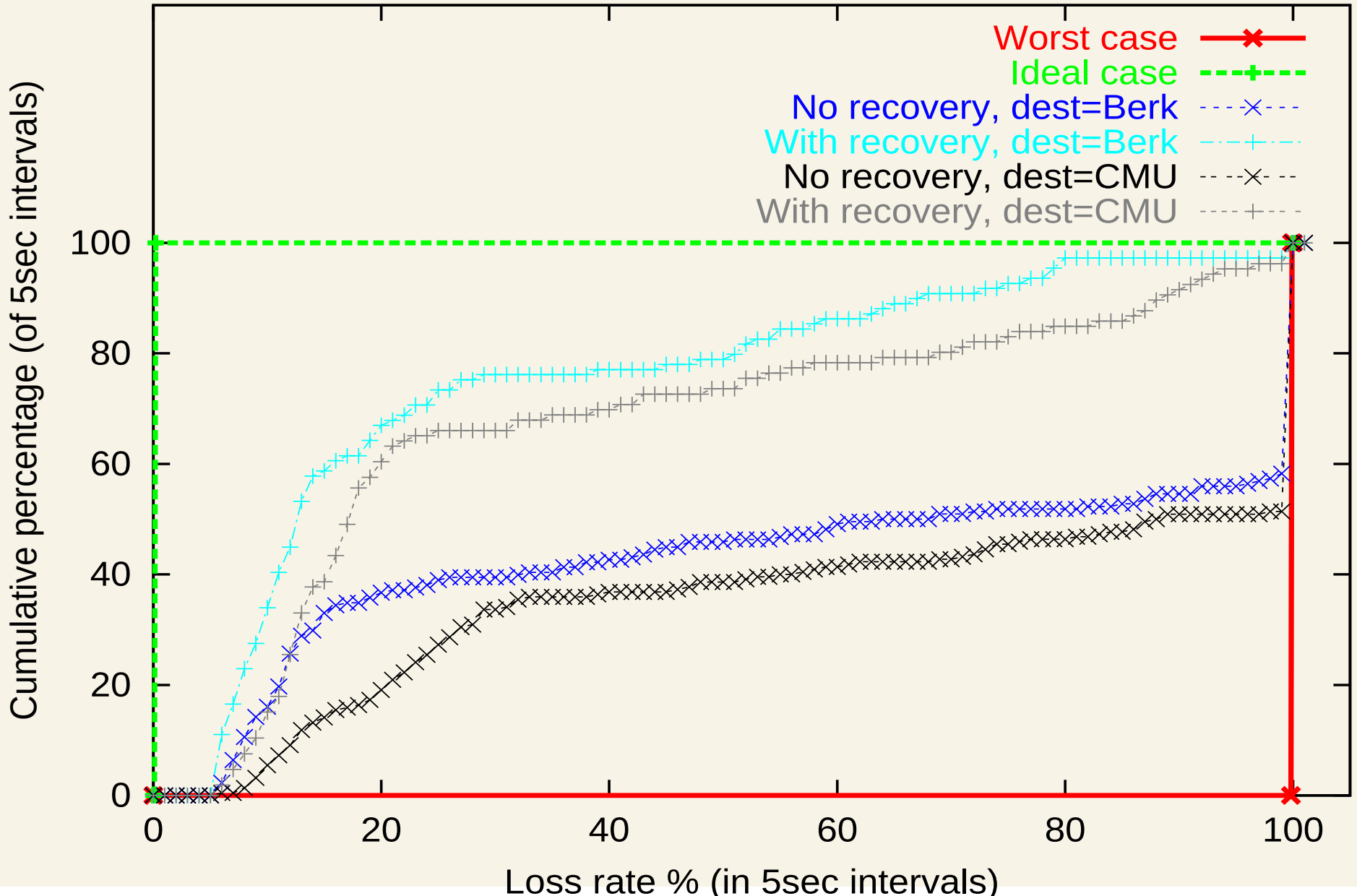
- Metric: loss-rate at end-client, measured in 5sec intervals

- Results shown at three levels:
  - Point of view of a pair of paths experiencing failure
  - Considering all failed paths
  - Overall improvement in availability

Result for a pair of paths



Result for all failed paths

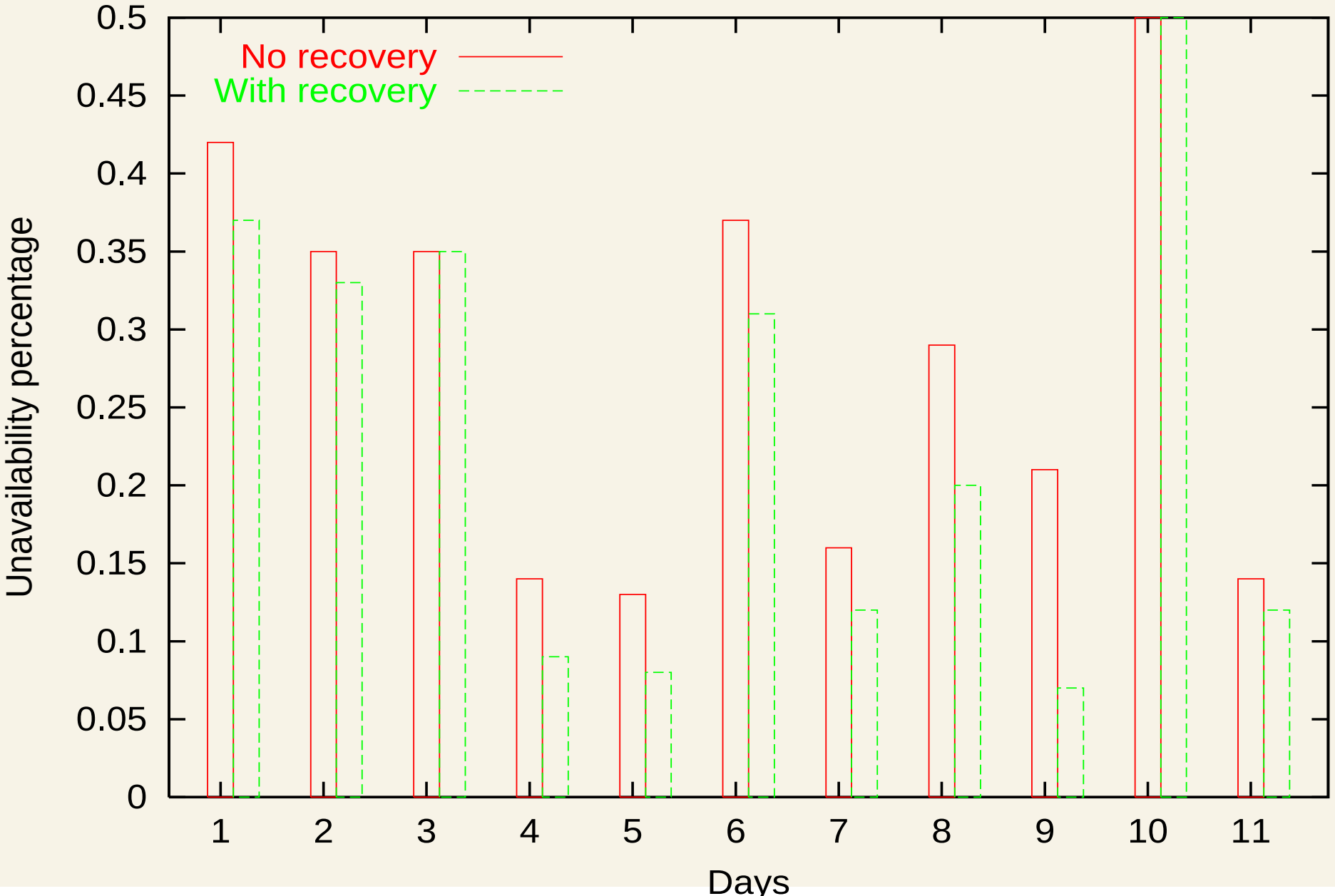


Improvement in availability

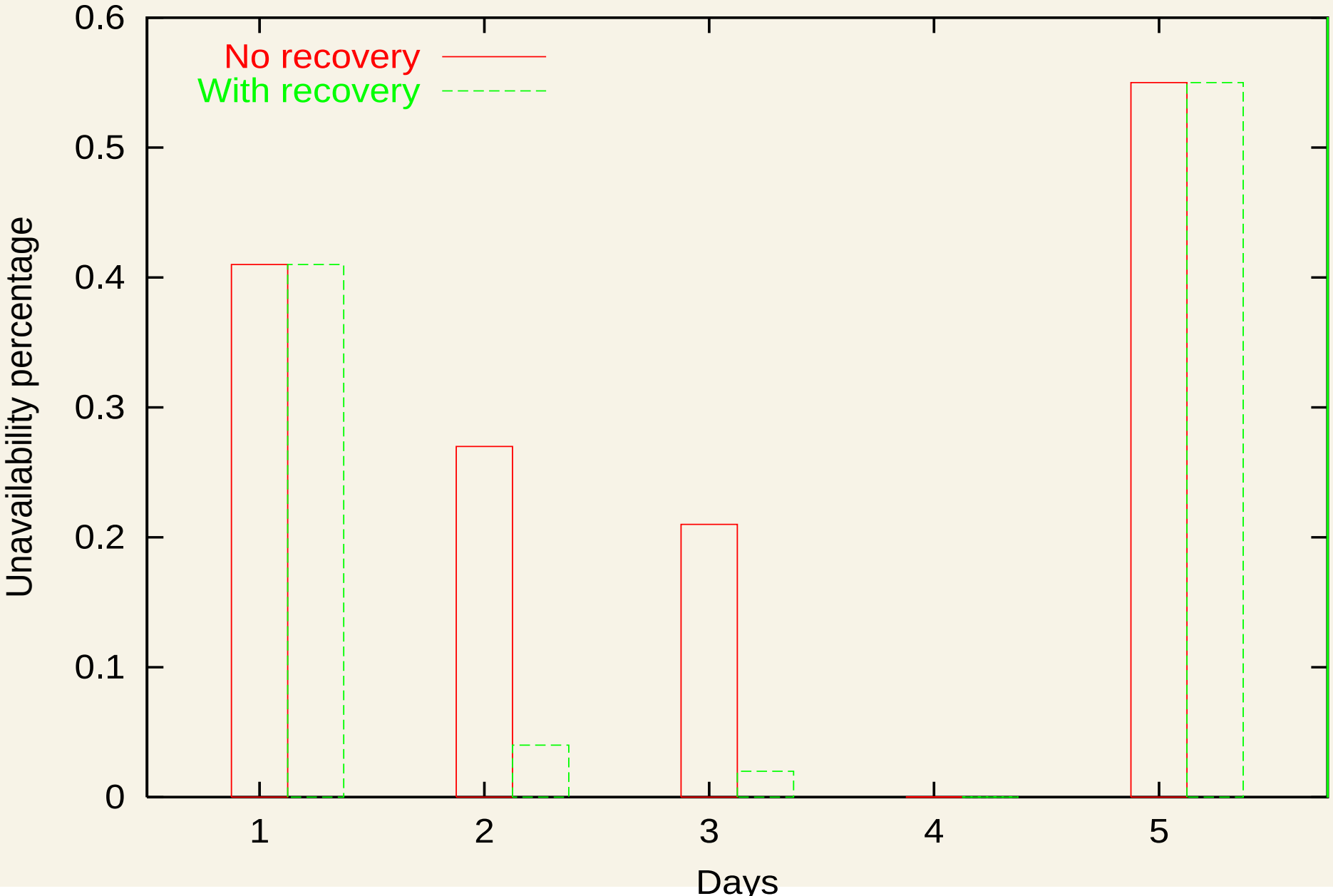
- Consider all paths: failed/recovered, not failed
  - There is a large percentage of the latter
- For each 5sec interval, define the system to have been available if
  - The loss-rate seen in that interval was < 20%
  - Otherwise, the system was unavailable for those 5sec
- Note: granularity of definition is 5sec, not the entire client session

- What is the improvement in availability due to recovery mechanism?

Availability Improvement (at Berkeley)



Availability Improvement (at CMU)



Some Remarks

- Failures near clients could not be masked in testbed setup
  - In a real setting with lots more service replicas, this would be possible
- We have measured the availability improvement due to the recovery algorithm alone
  - There are other components of the architecture
    - Link-state based dissemination of service reachability
    - Load-balancing metric (Infocom’03 paper)

Summary, and Future Directions

- Availability is harder as you get closer to 100.0%
  - Telephone networks have 99.999% availability
  - About 5min downtime in a year
- Significant improvements in availability through
  - Quick failure detection (2sec or less)
  - Architecture for quick failure recovery

- Interesting avenues for further exploration
  - Service composition can involve more complicated data paths
    - Data flow is a tree, not a path
  - Other challenges in multi-provider scenario
    - Some service replicas may not be usable, or may have higher cost