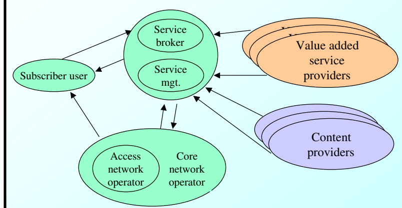
# An Architecture for Optimal and Robust Composition of Services across the Wide- Area Internet

Bhaskaran Raman
Qualifying Examination Proposal
Feb 12, 2001

Examination Committee:
Prof. Anthony D. Joseph (Chair)
Prof. Randy H. Katz
Prof. Ion Stoica
Prof. David Brillinger

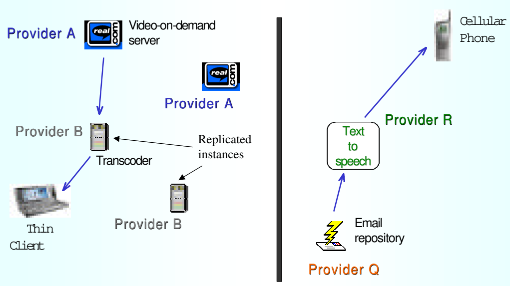
## Technological Trend

"Service and content providers play an increasing role in the value chain. The dominant part of the revenues moves from the network operator to the content provider. It is expected that value-added data services and content provisioning will create the main growth."




Access Networks
Cellular systems
Cordless (DECT)
Bluetooth
DECT data
Wireless LAN
Wireless local loop
Satellite
Cable
DSL

## Service Composition




***Reuse, Flexibility***

## Service Composition

- Operational model:
  - Service providers deploy different services at various network locations
  - Next generation portals compose services
    - Quickly enable new functionality on new devices
    - Possibly through SLAs
  - Code is NOT mobile (mutually untrusting service providers)
- Composition across
  - Service providers
  - Wide-area
- Notion of service-level path

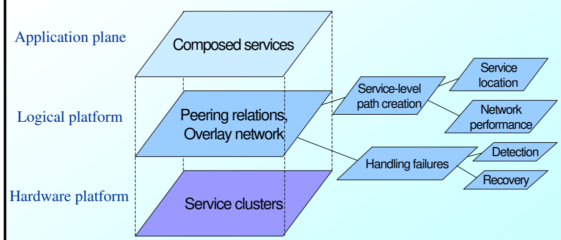
## Problem Statement

- Optimal service-level path creation
  - Based on network and service performance
- Robust service-level paths
  - Detect and recover from failures

## Challenges

- Optimal service-level path
  - When there are multiple instances of each intermediate service
  - How to learn the performance of each "leg" of the service-level path?
- Robustness
  - Detect and recover from failures
  - Possibly across the wide-area Internet
  - Important for long-lived sessions
    - Several minutes/hours
  - Quick recovery required for real-time applications
  - How to provide appropriate backup?

## Overall Architecture




## Problem Scope

- Services have no "hard" state
  - Sessions can be transferred from one service instance to another
  - This is assumed while handling failures
  - Assumption valid for a large set of applications
    - Content streaming
    - Transformation agents
    - Logical operations: personalized redirection

## Research Contributions

- Construction of optimal service-level path
  - Choice of instances of intermediate services based on network performance
- High availability for service-level paths
  - Mechanisms for detecting different kinds of failures
  - Creation of appropriate backup service-level path to recover from failures

## Outline

- Related work
- Feasibility of failure detection over the wide-area
- Design of the framework
- Evaluation
- Research methodology and timeline
- Summary

## Related work: Service Composition

- TACC (A. Fox, Berkeley)
  - Fault-tolerance within a single service-provider cluster for composed services
  - Based on cluster-manager/front-end based monitoring
- Simja (Berkeley), COTS (Stanford), Future Computing Environments (G. Tech)
  - Semantic issues addressed – which services can be composed
  - Based on service interface definitions, typing
- *None address wide-area network performance or failure issues for long-lived composed sessions*

## Related work: Performance and Robustness

- Cluster-based approaches: TACC, AS1, LARD
  - AS1 (E. Amir, Berkeley): soft-state model for maintenance of long-lived sessions
  - LARD (Rice Univ): Web-server load balancing within a cluster
  - Fault management and load balancing within a cluster
  - *Wide-area performance and failure issues not addressed*
- Wide-area server selection: SPAND (M. Stemm, Berkeley), Harvest (Colorado), Probing mechanisms
  - Network and/or server performance discovery for selecting optimal replica
  - *For composed services, require multi-leg measurement*
  - *For long-lived sessions, need recovery during session*
- Routing around failures: Tapestry/CAN (Berkeley), RON (MIT)
  - Use redundancy in overlay networks
  - *Recovery of composed service-level paths not addressed*

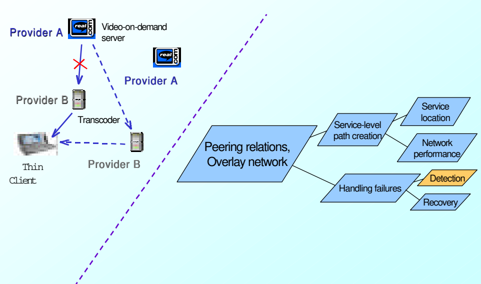
## Related work: summary

| | TACC | COTS, Future Comp. Env. | WA server selection | Tapestry, CAN | RON | Our System |
|---|---|---|---|---|---|---|
| Composed Services | Yes | Yes | No | No | No | Yes |
| WA perf. adaptation | No | No | Yes | ? | ? | Yes |
| Routing around failures | No | No | No | Yes | Yes | Yes |

## Outline

- Related work
- Feasibility of failure detection over the wide-area
- Design of the framework
- Evaluation
- Research methodology and timeline
- Summary

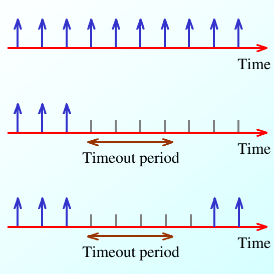
## Failure detection in the wide-area: Analysis




## Failure detection in the wide-area: Analysis

- What are we doing?
  - Keeping track of the liveness of the WA Internet path
- Why is it important?
  - 10% of Internet paths have 95% availability [Labovitz'99]
  - BGP could take several minutes to converge [Labovitz'00]
  - These could significantly affect real-time sessions based on service-level paths
- Why is it challenging?
  - Is there a notion of "failure"?
  - Given Internet cross-traffic and congestion?
  - What if losses could last for any duration with equal probability?

## Failure detection: the trade-off




Monitoring for liveness of path using keep-alive heartbeat

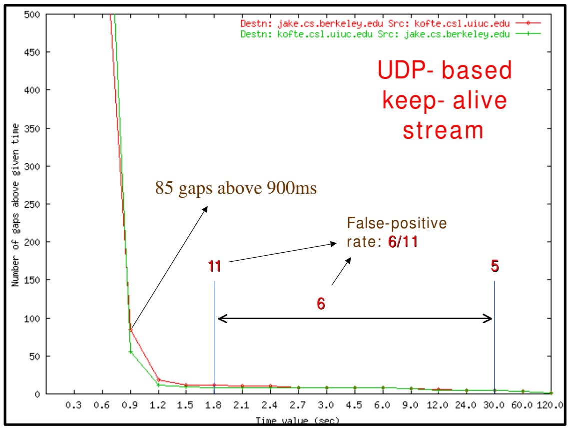
Failure: detected by timeout

False-positive: failure detected incorrectly Ë unnecessary overhead

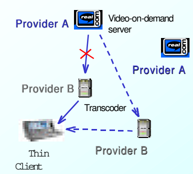
There's a trade-off between time-to-detection and rate of false-positives

## UDP-based keep-alive stream

- Geographically distributed hosts:
  - Berkeley, Stanford, UIUC, TU-Berlin, UNSW
  - Some trans-oceanic links, some within the US
- UDP heart-beat every 300ms between pairs
- Measure gaps between receipt of successive heart-beats




## UDP Experiments: What do we conclude?




- Significant number of outages > 30 seconds
  - Of the order of once a day
- But, 1.8 second outage Ë 30 second outage with 50% prob.
  - If we react to 1.8 second outages by transferring a session can have much better availability than what's possible today

## UDP Experiments: What do we conclude?

- 1.8 seconds good enough for non-interactive applications
  - On-demand video/audio usually have 5-10 second buffers anyway
- 1.8 seconds not good for interactive/live applications
  - But definitely better than having the entire session cut-off

## UDP Experiments: Validity of conclusions

- Results similar for other host-pairs:
  - Berkeley‡ Stanford, UIUC‡ Stanford, Berkeley‡ UNSW, TUBerlin‡ UNSW
- Results in parallel with other independent studies:
  - RTT spikes are isolated; undone in a couple of seconds [Acharya'96]
  - Correlation of packet losses does not persist beyond 1000ms [Yajnik'99]

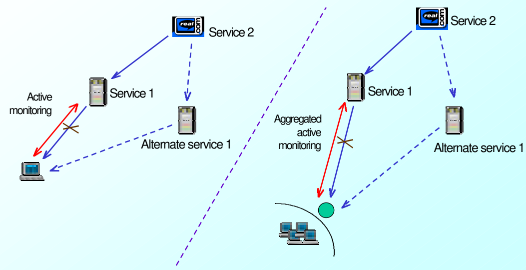
## Outline

- Related work
- Feasibility of failure detection over the wide-area
- Design of the framework
- Evaluation
- Research methodology and timeline
- Summary

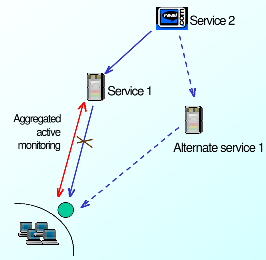
## Design of the Framework

- Question: how do we construct optimal, robust composed services?
- Need to:
  - Monitor for liveness
  - Monitor for performance
  - Compose services

## Design Alternative: Hop-by-hop composition

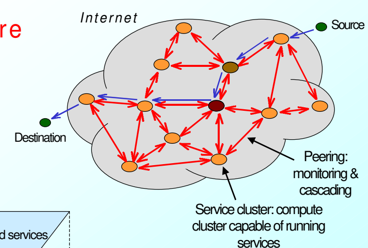


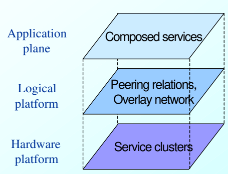

## Design Alternative: Hop-by-hop composition




- No special support for composition
- Problems:
  - Sub-optimal service-level path
  - Alternative path may not be active
  - What if both ends are fixed?

## Architecture

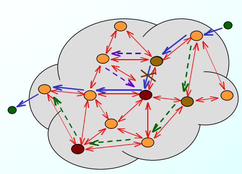


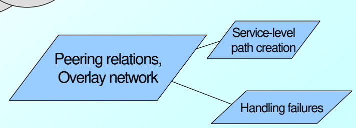

## Architecture: Advantages

- Overlay nodes are clusters
  - Compute platform for services
  - Hierarchical monitoring
    - Within cluster – for process/machine failures
    - Across clusters – for network path failures
  - Aggregated monitoring
    - Amortized overhead

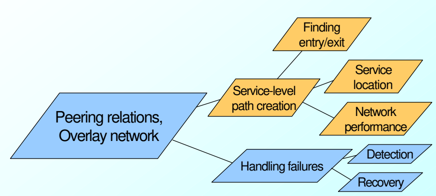
## The Overlay Network



The overlay network provides the context for service-level path creation and failure handling




## Service Level Path Creation



## Finding entry and exit

- Independent of other mechanisms
- We do not address this directly
- We assume:
  - Entry or exit point can be rather static
    - By appropriate placement of overlay nodes
    - Pre-configuration possible at end-host
  - Or, can learn entry or exit point through another level of indirection

## Service- Level Path Creation

- Connection-oriented network
  - Explicit session setup stage
  - There's "switching" state at the intermediate nodes
- Need a connection-less protocol for connection setup
- Need to keep track of three things:
  - Network path liveness
  - Metric information (latency/bandwidth) for optimality decisions
  - Where services are located

## Service- Level Path Creation

- Three levels of information exchange
  - Network path liveness
    - Low overhead, but very frequent
  - Metric information: latency/bandwidth
    - Higher overhead, not so frequent
    - Bandwidth changes only once in several minutes [Balakrishnan'97]
    - Latency changes appreciably only once in about an hour [Acharya'96]
  - Information about location of services in clusters
    - Bulky, but does not change very often (once in a few weeks, or months)
    - Could also use independent service location mechanism

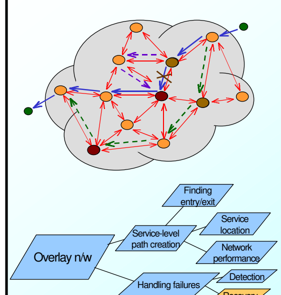
## Service Level Path Creation

- Link-state algorithm to exchange information
  - Lesser overhead of individual measurement Ë finer time-scale of measurement
  - Service-level path created at entry node
  - Link-state because it allows all-pair-shortest-path calculation in the graph

## Service Level Path Creation

- Two ideas:
  - Path caching
    - Remember what previous clients used
    - Another use of clusters
  - Dynamic path optimization
    - Since session-transfer is a first-order feature
    - First path created need not be optimal

## Session Recovery: Design Tradeoffs




- End-to-end vs. local-link
- End-to-end:
  - Pre-establishment possible
  - But, failure information has to propagate
  - And, performance of alternate path could have changed
- Local-link:
  - No need for information to propagate
  - But, additional overhead

## The Overlay Topology: Design Factors

- How many nodes?
  - Large number of nodes Ë lesser latency overhead
  - But scaling concerns
- Where to place nodes?
  - Close to edges so that hosts have points of entry and exit close to them
  - Close to backbone to take advantage of good connectivity
- Who to peer with?
  - Nature of connectivity
  - Least sharing of physical links among overlay links

## Outline

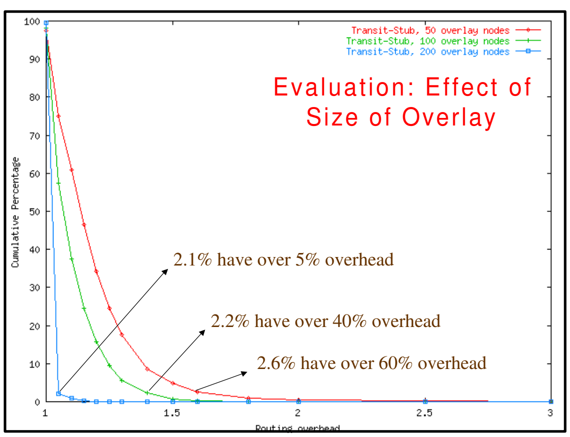
- Related work
- Feasibility of failure detection over the wide-area
- Design of the framework
- Evaluation
- Research methodology and timeline
- Summary

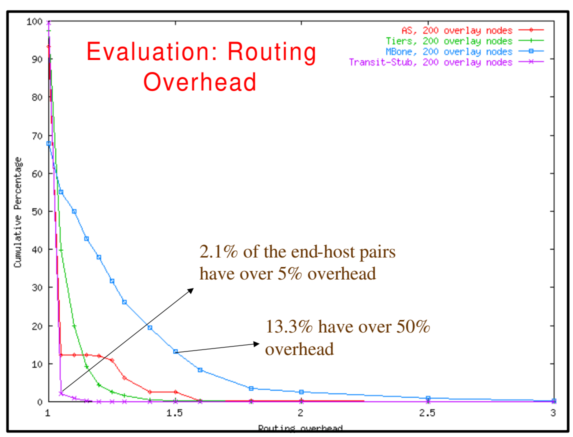
## Evaluation

- Important concerns:
  - Overhead of routing over the overlay network
  - Number of overlay nodes required
- Network modeling
  - AS-Jan2000, MBone, TIERS, Transit-Stub
  - Between 4000-6500 nodes
  - Each node represents an Address Prefix [Jamin'00]
  - In comparison, the Internet has ~100,000 globally visible APs [Jamin'00]

## Evaluation

- Overlay nodes
  - 50, 100, 200: those with max. degree (backbone placement)
  - Peering between "nearby" overlay nodes
  - Physical links are not shared
- 1000 random pairs of hosts in original network
  - Overhead of routing over overlay network
  - No intermediate services used – for isolating the raw latency overhead



## Evaluation: What can we conclude?

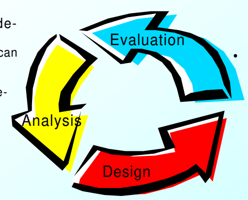
- Number of overlay nodes required for low latency quite low:
  - 200 for 5000 nodes
  - How many for 100,000 nodes? (number of APs on the Internet)
  - For linear growth, 4000 overlay nodes (in comparison, there are ~10,000 ASs on the Internet)
  - Note: growth has to be sub-linear
- Latency overhead of using overlay network quite low
  - Can get away with < 5% overhead in most nodes in some cases

## Outline

- Related work
- Feasibility of failure detection over the wide-area
- Design of the framework
- Evaluation
- Research methodology and timeline
- Summary

## Research Methodology

- Wide-area monitoring trade-offs
  - How quickly can failures be detected?
  - Rate of false-positives




- Simulation
  - Routing overhead
  - Effect of size of overlay
- Implementation
  - MP3 music for GSM cellular-phones
  - Codec service for IP-telephony

- Connection-oriented overlay network of clusters
- Session-transfer on failure
- Aggregation – amortization of overhead

## Research Methodology: Metrics

- Feasibility:
  - Overhead
    - End-to-end latency; bandwidth for information exchange
  - Scalability
    - To a large number of client sessions
  - Stability
    - Of optimality and session-transfer decisions
- Usefulness:
  - Latency to recovery
    - Measure of effectiveness
  - Use of composability
    - For building application functionality

## Research Methodology: Approach

- Simulations, Trace-collection, Real implementation
- Simulation
  - For initial estimation of overhead
- Simulation + Traces
  - Bandwidth usage estimation, Stability study
- Real implementation
  - Scalability studies
  - Real services for use of composability
- Testbed
  - Collaborating with UNSW, TUBerlin

## Research Plan: Phase I (0- 6 months)

- Detailed analysis of
  - Latency and bandwidth overhead
  - Latency to recovery
- Use traces of latency/bandwidth over wide-area
- Develop real implementation in parallel
  - This is already in progress
  - Will give feedback for the analysis above

## Research Plan: Phase II (6- 12 months)

- Use implementation from Phase I
  - Deploy real services on the wide-area testbed
  - Analyze end-to-end effects of session-recovery
  - Examine scalability
- Use traces from Phase I to analyze stability of optimality decisions
  - Collect more traces of latency/bandwidth

## Research Plan: Phase III (12- 18 months)

- Use feedback from deployment of real services to refine architecture
- Analyze placement strategies
  - Use wide-area measurements and traces from phases I and II
- Write, graduate…

Appropriate conferences and workshops:
NOSSDAV, ACM Multimedia, SOSP, INFOCOM,
SIGMETRICS, SIGCOMM

## Summary

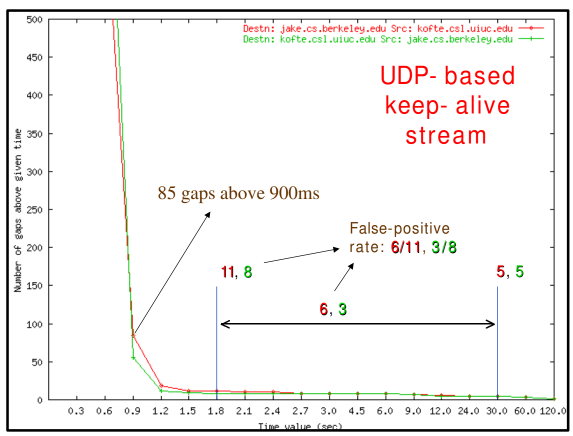
- Logical overlay network of service clusters
  - Middleware platform for service deployment
  - Optimal service-level path creation
  - Failure detection and recovery
- Failures can be detected in O(1sec) over the wide-area
  - Useful for many applications
- Number of overlay nodes required seems reasonable
  - O(1000s) for minimal latency overhead
- Several interesting issues to look at:
  - Overhead, Scalability, Stability

## References


- [Labovitz'99] C. Labovitz, A. Ahuja, and F. Jahanian, *"Experimental Study of Internet Stability and Wide-Area Network Failures"*, Proc. Of FTCS'99
- [Labovitz'00] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, *"Delayed Internet Routing Convergence"*, Proc. SIGCOMM'00
- [Acharya'96] A. Acharya and J. Saltz, *"A Study of Internet Round-Trip Delay"*, Technical Report CS-TR-3736, U. of Maryland
- [Yajnik'99] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, *"Measurement and Modeling of the Temporal Dependence in Packet Loss"*, Proc. INFOCOM'99
- [Balakrishnan'97] H. Balakrishnan, S. Seshan, M. Stemm, and R. H. Katz, *"Analyzing Stability in Wide-Area Network Performance"*, Proc. SIGMETRICS'97


## Related work: Routing around Failures

- Tapestry, CAN (Berkeley)
  - Locate replicated objects in the wide-area using an overlay network
  - Overlay nodes named such that routing table is small
  - Redundancy in the overlay network helps in availability in the presence of failures
- Resilient Overlay Networks (MIT)
  - Small number (~50) of nodes on the Internet form a redundant overlay network
  - Application level routing metrics, and quick recovery from failures
- *Recovery of composed service-level paths not addressed*



## Open Issues

- Using application level information for dynamic path construction
  - Some transformations may not be good
- Service interfaces for automated composition and correctness checking
- Soft-state based management of application level state

## UDP- based keep- alive stream

| HB destination | HB source | Total time | Num. False positives | Num. Failures |
|---|---|---|---|---|
| Berkeley | UNSW | 130:48:45 | 135 | 55 |
| UNSW | Berkeley | 130:51:45 | 9 | 8 |
| Berkeley | TU-Berlin | 130:49:46 | 27 | 8 |
| TU-Berlin | Berkeley | 130:50:11 | 174 | 8 |
| TU-Berlin | UNSW | 130:48:11 | 218 | 7 |
| UNSW | TU-Berlin | 130:46:38 | 24 | 5 |
| Berkeley | Stanford | 124:21:55 | 258 | 7 |
| Stanford | Berkeley | 124:21:19 | 2 | 6 |
| Stanford | UIUC | 89:53:17 | 4 | 1 |
| UIUC | Stanford | 76:39:10 | 74 | 1 |
| Berkeley | UIUC | 89:54:11 | 6 | 5 |
| UIUC | Berkeley | 76:39:40 | 3 | 5 |

## Example services that can be composed

- Content streaming
  - Audio/Video
- Transcoding
  - Format translation
  - Text to speech
- Rate adaptation
  - Lossy encoding
  - Reduction of frame size
- Adding FEC

## Example services that can be composed

- Unicast to multicast and vice-versa
- Personalized redirection
  - Between multiple user devices
  - Service handoff
- Adding semantic content
  - E.g., song title, or classification
  - Ads L

## Some more composed services

- MP3 ‡ PCM ‡ GSM
- MP3 ‡ Reduce quality ‡ Add FEC for wireless link
- Video ‡ Redirector (Handheld/Desktop)

## Link State vs. Distance Vector

- Link State:
  - Quicker reaction to failures
  - Failure information need not propagate
  - Multiple metrics possible (app level)
  - Important reason: need distances from intermediate nodes for composition
- Distance Vector requires lesser storage
  - Not true with path vector
- Why not Path Vector?
  - Convergence could take a long time [Labovitz'00]