

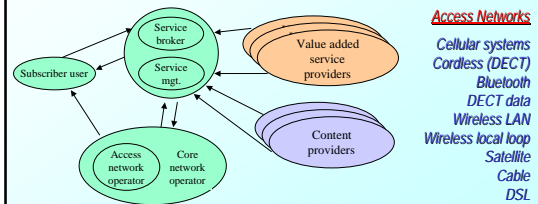
A Framework for Composing Services Across Independent Providers in the Wide-Area Internet

Bhaskaran Raman
Qualifying Examination Proposal
Feb 12, 2001

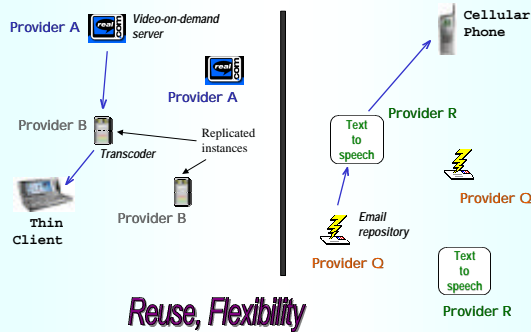
Examination Committee:
Prof. Anthony D. Joseph (Chair)
Prof. Randy H. Katz
Prof. Ion Stoica
Prof. David Brillinger

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."



Service Composition



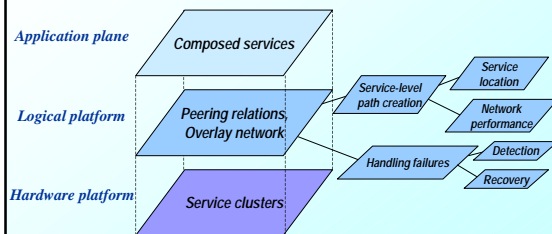
Service Composition

- 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 [Roscoe00]
- Composition across
 - Service providers
 - Wide-area
- Notion of **service-level path**

Requirements and Challenges

- Framework for composing services
 - How are services deployed/replicated?
 - Who composes services? How are service-level paths created?
- Choice of "optimal" service-level path
 - When there are multiple instances of each intermediate service
- 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

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 [Snoeren01, Brassil01]
 - Content streaming
 - Transformation agents
 - Addition of semantic content (e.g., song title)
 - Logical operations: redirection

Research Contributions

- Framework for composing services
 - Optimality - choice of service instances
 - High availability - failure detection and recovery
- Develop applications that use such composition
 - Demonstrate use of mechanisms for optimality and failure recovery

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, strict typing
- HP e-speak
 - Service description and discovery model
 - Scalability?
- *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
 - Fault management and load balancing within a cluster
 - *Wide-area performance and failure issues not addressed*
- Wide-area server selection: SPAND, Harvest, 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*

Related work: Routing around Failures

- Tapestry, CAN
 - Locate replicated objects in the wide-area using an overlay network
 - Redundancy in the overlay network helps in availability in the presence of failures
- Resilient Overlay Networks
 - 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*

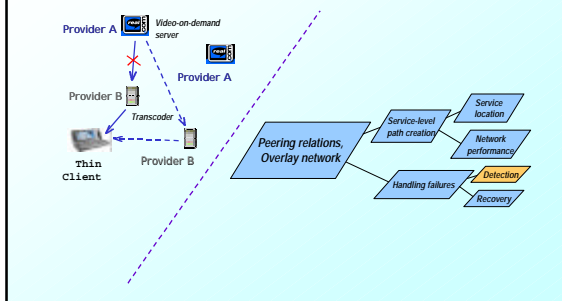
Related work: summary

	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

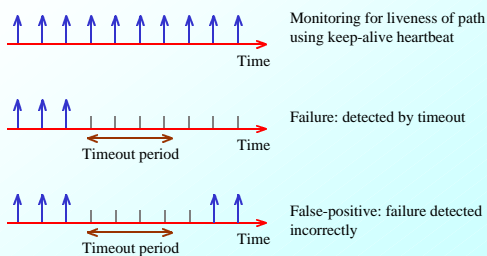
Failure detection in the wide-area: Analysis



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 [IPMA1]
 - BGP could take several minutes to converge [IPMA2]
 - 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?

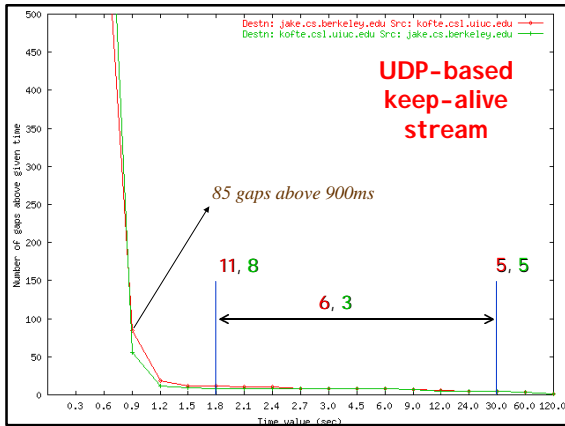
Failure detection: the trade-off



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
 - Choice of time value justified later...
- 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
 - Availability much lesser than in PSTN
 - Along the lines of findings in [IPMA1]
- But, 1.8 second outage → 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
- 1.8 seconds not good for interactive/live applications
 - But definitely better than having the entire session cut-off
 - May require further application support

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 [AS96]
 - 86% of bad TCP timeouts are due to one or two elevated RTTs [AP99]
 - Correlation of packet losses does not persist beyond 1000ms [Yajnik98]

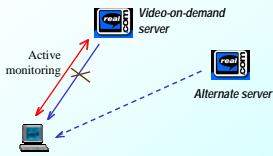
Outline

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

Design of the Framework

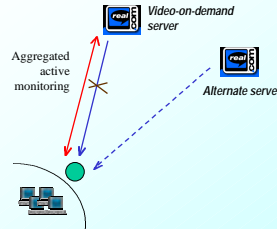
- Question: how do we construct optimal, robust composed services?

Design Alternative: End-to-end monitoring



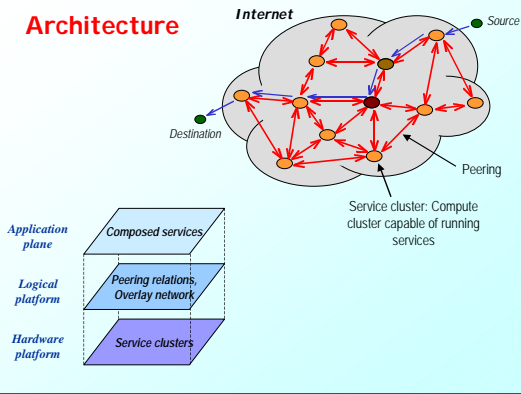
- No infrastructure required
 - Hop-by-hop composition
- Problems:
 - Overhead
 - Sub-optimal service-level path
 - Alternative path may not be active
 - What if both ends are fixed?

Design Alternative: Client-Side Aggregation



- Reduces overhead
- Other problems persist:
 - Hop-by-hop composition
 - Alternate server could be unavailable
 - Does not work if both ends are fixed

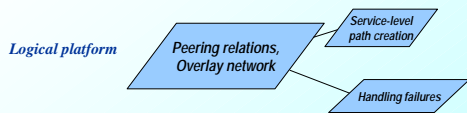
Architecture



Architecture: Advantages

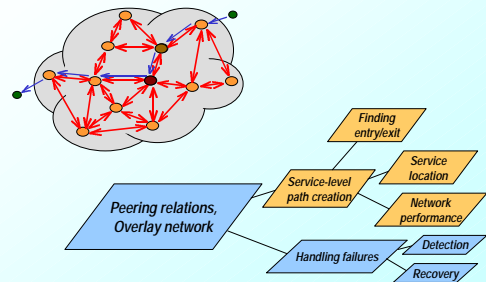
- Overlay nodes are clusters
 - Hierarchical monitoring
 - Within cluster - for process/machine failures
 - Across clusters - for network path failures
 - Aggregated monitoring
 - Amortized overhead
- Overlay network
 - Intuitively, expected to be much smaller than the Internet
 - With nodes near the backbone, as well as near edges

Architecture: 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
- Entry or exit point can be rather static
 - Nodes are clusters → do not fail often
 - By placement, can make choice of overlay node obvious
- Can learn entry or exit point through
 - Pre-configuration,
 - Expanding scope search,
 - Or, any other 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

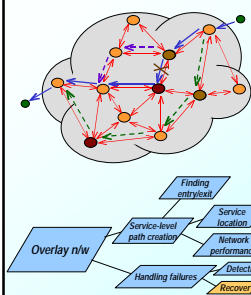
Service-Level Path Creation

- 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 [SPAND]
 - Latency changes appreciably only once in about an hour [AS96]
 - Information about location of services in clusters
 - Bulky
 - But does not change very often (once in a few weeks, or months)
- Link-state algorithm to exchange information
 - Least overhead → max. frequency
- Service-level path created at entry node

Routing on the overlay network

- 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
 - Pre-established vs. on-demand
 - Can use a mix of strategies
- Pre-established end-to-end:
 - Quicker setup of alternate path
 - But, failure information has to propagate
 - And, performance of alternate path could have changed
- On-demand local-link:
 - No need for information to propagate
 - But, additional overhead

The Overlay Topology

- Need to address:
 - How many overlay nodes are deployed?
 - Where are they deployed?
 - How do they decide to peer?

The Overlay Topology: Design Factors

- How many nodes?
 - Large number of nodes → lesser latency overhead
 - But scaling concerns
- Where to place nodes?
 - Need to have overlay nodes close to edges
 - Since portion of network between edge and closest overlay node is not monitored
 - Need to have overlay nodes close to backbone
 - Take advantage of good connectivity
- Who to peer with?
 - Nature of connectivity
 - Least sharing of physical links among overlay links

Outline

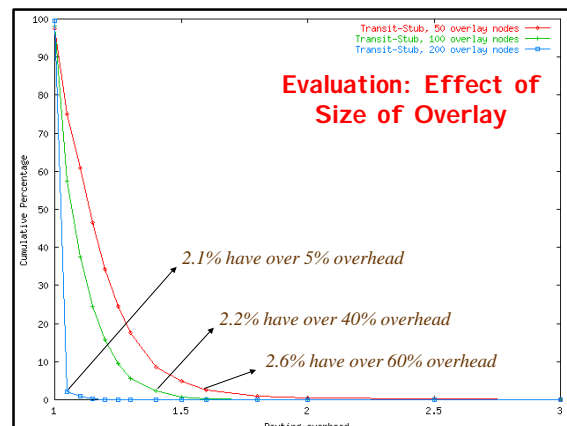
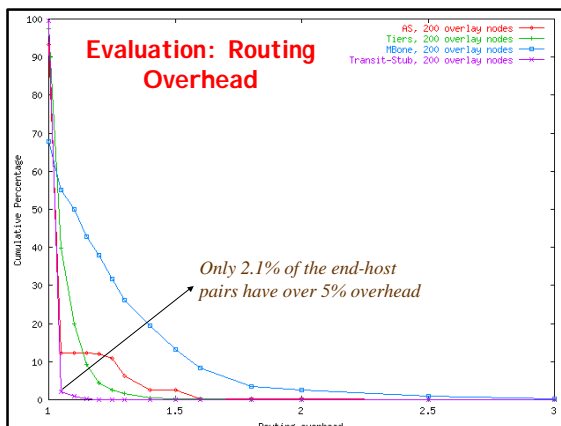
- Related work
- Feasibility of failure detection over the wide-area
- Design of the framework
- Evaluation
- Research methodology and timeline
- Summary

Evaluation

- Important concern: overhead of routing over the overlay network
 - Addition to end-to-end latency
- Network modeling
 - AS-Jan2000, Mbone, TIERS, Transit-Stub
 - Between 4000-6500 nodes
 - Each node represents an Address Prefix [IDMaps]
 - In comparison, the Internet has ~100,000 globally visible APs [IDMaps]

Evaluation

- Overlay nodes
 - 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?

- Latency overhead of using overlay network quite low
 - Can get away with < 5% overhead in most cases
- 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)

Outline

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

Research Methodology

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

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, SOS, INFOCOM,
SIGCOMM*

Summary of Contributions

- Framework for service composition across service providers
- Notion of connection-oriented network at the service-level
 - For optimizing paths
 - For detecting and recovering from failures