6.829 Overlays, P2P and Application Multicast

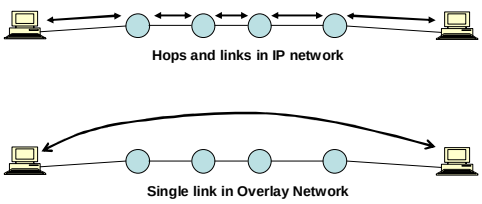
Rob Beverly <rbeverly@mit>
November 9, 2006

Overlays

Overlays

- Network:
  - An addressing, routing and service model for communication between hosts
- Overlay:
  - Network built on top of another network
  - Adds layer of indirection/virtualization
  - Offer different properties than underlying network
  - One “hop” in overlay may be many hops in underlay
  - One link in IP network is between two routers
  - One link in overlay is between two hosts in IP network

Overlays



Overlays (Dis)advantages

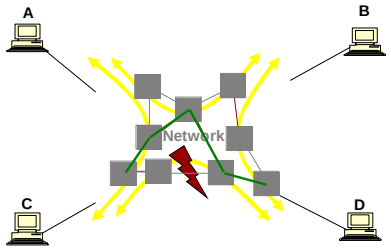
- Advantages:
  - Rapidly deploy new services, no innovation barrier
  - No new protocols
  - No new equipment
  - Incremental deployment
  - E.g. IP over Ethernet, does not require modifying Ethernet protocol, driver, etc
- Disadvantages:
  - Adds overhead, additional layers
  - Adds complexity
  - Unintentional interaction between new layers of abstraction, e.g. TCP reacting to wireless loss

What’s an Overlay?

- Internet is/was an overlay on top of phone network
- Single router “hop” may involve many links via lower-layer TE, e.g. MPLS
- Vonage is an VoIP over Internet over POTS overlay?
- Some examples:
  - CDNs (e.g. Akamai)
  - Peer-to-Peer (P2P)
  - Mobility
  - Mbone (Connect islands of multicast)
  - 6bone (IPv6 deployment stop-gap)
  - Anonymity (Tor, onion routing, others)

Internet Routing

Internet Abstraction



Any-to-any communication, routing around failures

Internet Routing

- Often many physical paths between hosts
- No *user* control
- Source routing via IP options not viable
  - Economics of user-directed routing interesting research question
- Routing not load or loss sensitive
- Routing updates are damped
- Convergence time can be long
- MRAI timer, Policy from SIGCOMM06 paper

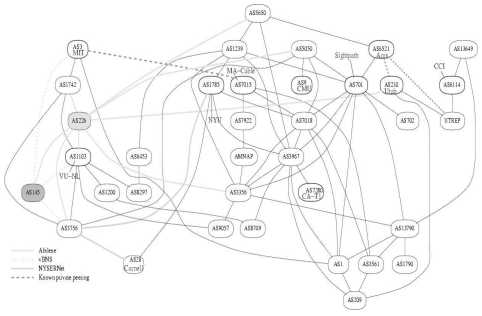
Types of Failures

- Path failure
  - Configuration / operational errors
  - Software error
- Performance failure
  - Congestion
  - Denial of service
  - Large delays

Measurements of Internet Performance

Paxson 95-97	• 3.3% of all routes had serious problems
Labovitz 97-00	• 10% of routes available < 95% of the time • 65% of routes available < 99.9% of the time • 3-min minimum detection+recovery time; often 15 mins • 40% of outages took 30+ mins to repair
Chandra 01	• 5% of faults last more than 2.75 hours

Redundant Paths in Internet

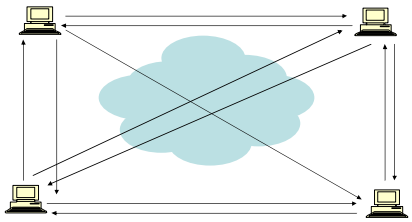


Resilient Overlay Networks

RON Approach

- Cooperating hosts in different routing domains can forward traffic for each other
- Detect failures faster than Internet routing
- Route around failures
- Achieve better paths
- Assumption: small O(10) hosts

RON Path Probing

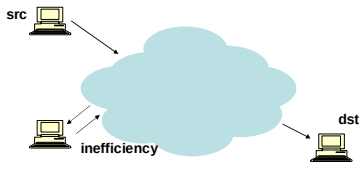


- Frequently measure all node paths
- Exchange routing information
- Route along best path

Does RON Work?

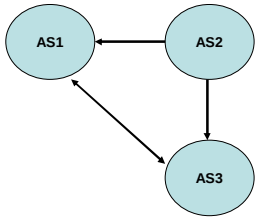
- Questions:
  - Does RON violate ISP policies?
  - Is routing stable?
  - How many outages are avoidable?
  - How distributed are path failures? All over
  - How near are path failures? Multihomed hosts.
  - Can route around failure before failure event is over?
  - Does detecting failure require active probing?
- Advantages?
  - Better than wide-area protocols?
- Disadvantages?
  - Scalability? O(N^2) probing?
  - Probing itself can introduce congestion?

RON Indirection Efficiency



- In and out of host interface, same traffic on link twice
- Internet as RON: every packet sent twice
- Many RONs? Tragedy of the commons?
- Drafting behind Akamai SIGCOMM06 paper: using CDN infrastructure measurements to inform other overlays

Overlays Altering Business Relationships

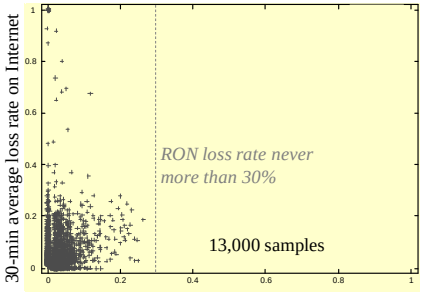


AS1 to AS2 via customer-provider link
What about overlay that drives AS1 traffic to AS3 to AS2?

RON Evaluation Methodology

- 19 node deployment
- Repeat:
  - Pick random node j
  - Pick probe type round robin from {direct, latency, loss}
  - Delay for random interval [1-2]seconds

Packet Loss over 30 minute Intervals



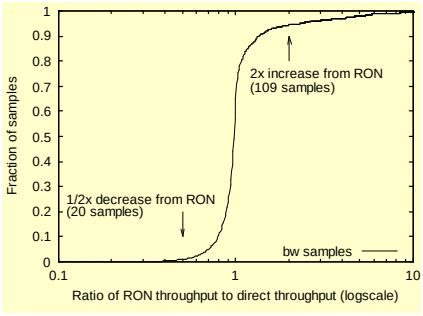
RON Failure Improvement

30-minute average loss rates

Loss Rate	RON Better	No Change	RON Worse
10%	479	57	47
20%	127	4	15
30%	32	0	0
50%	20	0	0
80%	14	0	0
100%	10	0	0

6,825 "path hours" represented here
12 "path hours" of essentially complete outage
76 "path hours" of TCP outage
RON routed around all of these!
One indirection hop provides almost all the benefit!

RON Throughput Improvement



Peer-to-Peer

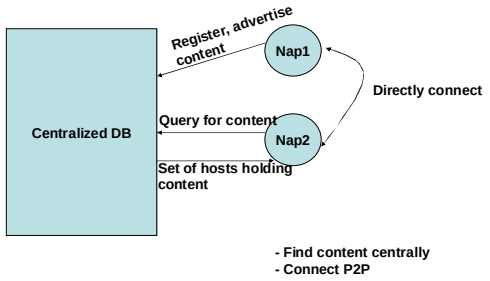
Peer-to-Peer

- No longer a client-server model
- Hosts are both clients and servers
- Alleviate load on single server
- Self-scaling networks:
  - Aggregate bandwidth proportional to number of members

Napster

- Peers connect to central database that maintains per-peer:
  - Connection state
  - Available content
- Peer search for content by querying database
- Database has complete "view" of network
- Main insight:
  - Separate finding content from obtaining content
  - Hosts are both clients and servers
- Downfall:
  - Centralized

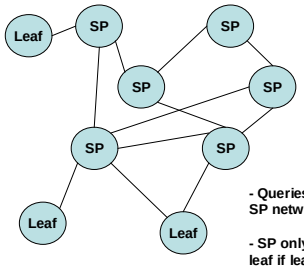
Napster



Gnutella

- Distributed search via flooding
- Unstructured network formation, organic
- Flooding problem addressed by hierarchy
- Two-level hierarchy of SuperPeers (SP) and Leafs:
  - SP to SP
  - Leafs to SP
- SP generally have high-bandwidth, long-lived
- Queries flooded through SP network with limited TTL horizon
- SP knows what content leafs have
  - Efficient bloom filter representation
- Once peer with content is found, peers connect directly; overlay only used to *find* content
- If content exists in system will a query always find the peer offering that content?

Gnutella



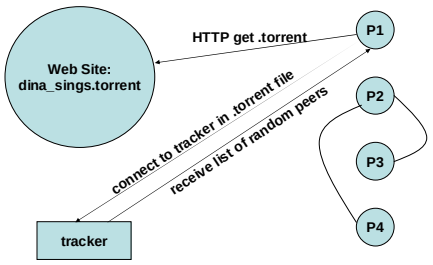
DHT

- Next lecture: Distributed Hash Tables
- Another type of overlay
- Offer guaranteed lookups in upper bounded lookup complexity
- e.g. chord from MIT, O(*logn*) lookup hops in overlay, O(*logn*) state maintained per node

Bittorrent

- Overlay, doesn't solve finding content problem
- Main insights:
  - Advertise content via HTTP link to torrent "tracker"
  - Centralized per-torrent tracker keeps track of peers
  - Split file into chunks
  - Fairness: upload rate proportional to download rate

Bittorrent



Bittorrent

- Trackers only help peers find each other
- Trackers return random set of peers download the same file:
  - Construct robust connection graph in the face of churn
- Peers selfishly attempt to maximize their download rate
- Files are chunked and torrent contains SHA hashes of each chunk for verification
- RIAA attacks: malicious bittorrent nodes

Bittorrent

- Piece selection:
  - Rarest first. Why?
  - Random first piece. Why?
  - Endgame mode. Why?
- Choking:
  - Unchoke peers that are uploading to us
  - Optimistically trial choked connections
  - Download rate: 20 sec rolling average
- Download finished:
  - Maintain highest rate uploads
  - Upload to peers no one else is uploading to

Bittorrent

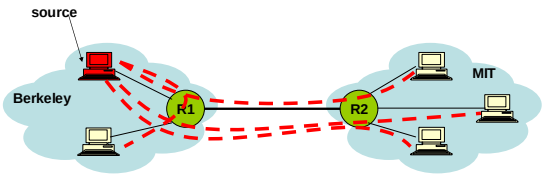
- Interesting economic spin on networking problem
- Deployed, highly successful
- Questions:
  - Again centralized peer finding. Do better?
  - Are users altruistic?
  - Is user cost=0 until congestion (step function?)
  - Gaming bittorrent? (cheap pseudonyms)

Application Layer Multicast

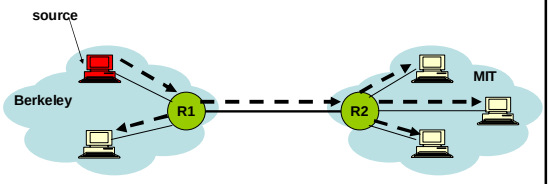
Multicast

- Problem:
  - Single source, multiple (*N*) receivers
- Options:
  - Source sends *N* individual streams
  - Network routers maintain distribution tree of interested parties and replicate packets
  - Hosts form an overlay tree and replicate packets
- Focus on overlay solution: application layer multicast

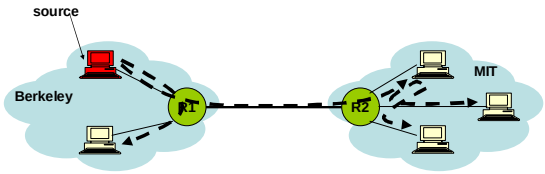
Unicast Replication



Network Multicast



Application Layer Multicast



Metrics to Evaluate Multicast

- Quality of Delivery Path:
  - *Stress*: per-link number of times an identical packet is seen
  - *Stretch*: per-member ratio of path length in overlay (from source to member) to unicast (direct) path length
  - *Node degree*: larger degree=more load, fairness issues
- Overlay Robustness
- Control Traffic

Metrics of Unicast Solution

- Example: unicast to each receiver
  - Minimizes stretch, stretch = 1
  - Stress: O(*N*) near source
  - Robustness? Great – single failure doesn't affect other members

NICE

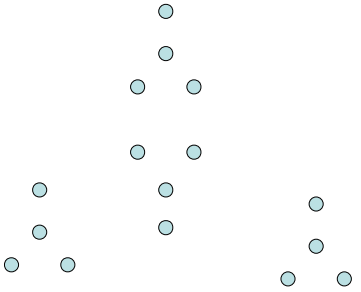
NICE

- Recursive acronym: NICE is Internet Cooperative Environment
- Assign members to layers starting at L_0
- Hosts in each layer partitioned into set of clusters
- Cluster size: *k*
- Each cluster has cluster leader which is graph-theoretic center:
  - Minimize maximum distance

NICE

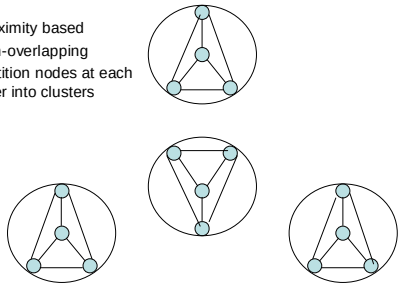
- All hosts are part of lowest layer L_0
- Cluster leaders of layer L_i join layer L_{i+1}
- Hierarchical clustering
- Highest layer has only a single member
- log(n) layers

NICE

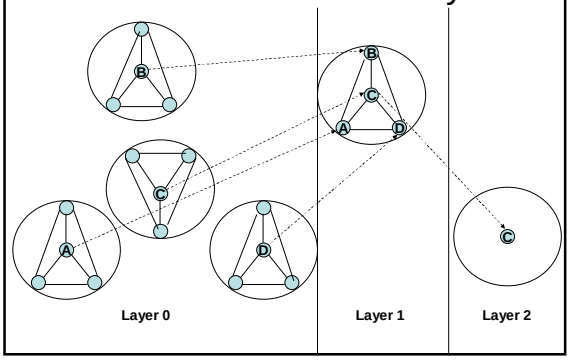


NICE Clusters

- Proximity based
- Non-overlapping
- Partition nodes at each layer into clusters



NICE Cluster Hierarchy



NICE Data Tree

- Host h with a packet from p:
- If h is cluster leader, send to peers in next higher layer
- If p from host not in cluster, send to all nodes of current cluster
- Control topology is a clique
- Data topology is a star tree

NICE Join Procedure

- Join via a rendezvous point (RP)
- RP gives address of node in highest layer
- Each join message gives addresses of cluster leaders in next lower layer
- Send subsequent join to lowest latency peer
- Each join has successively lower latency, latency variance
- See paper for more details

NICE Guarantees

- With *k* sized clusters
- A node in L_0 will peer with O(*k*) other nodes
- A peer in L_i will peer with O(*k*) in each of the *i* levels: O(*ki*)
- Highest level: O(*klogN*), gives us worst case control overhead
- Same analysis for measuring stress