CS5412: USING GOSSIP TO BUILD OVERLAY NETWORKS
Gossip and Network Overlays

- A topic that has received a lot of recent attention
- Today we’ll look at three representative approaches
  - Scribe, a topic-based pub-sub system that runs on the Pastry DHT (slides by Anne-Marie Kermarrec)
  - Sienna, a content-subscription overlay system (slides by Antonio Carzaniga)
  - T-Man, a general purpose system for building complex network overlays (slides by Ozalp Babaoglu)
Research done by the Pastry team, at MSR lab in Cambridge England

Basic idea is simple

- Topic-based publish/subscribe
- Use topic as a key into a DHT
  - Subscriber registers with the “key owner”
  - Publisher routes messages through the DHT owner

Optimization to share load

- If a subscriber is asked to forward a subscription, it doesn’t do so and instead makes note of the subscription. Later, it will forward copies to its children
Architecture

Scalable communication service

TCP/IP

Internet

PASTRY

DHT

P2P location and routing layer

subscription management
Event notification

SCRIBE
Design

- Construction of a multicast tree based on the Pastry network
  - Reverse path forwarding
  - Tree used to disseminate events
- Use of Pastry route to create and join groups
SCRIBE: Tree Management

- Create: route to groupId
- Join: route to groupId
- Tree: union of Pastry routes from members to the root.
- Multicast: from the root down to the leaves

- Low link stress
- Low delay

Forwards two copies

join(groupId)

Multicast (groupId)
SCRIBE: Tree Management

Name space

Proximity space

CS5412, Spring 2016
Concerns?

- Pastry tries to exploit locality but could these links send a message from Ithaca... to Kenya... to Japan...
- What if a relay node fails? Subscribers it serves will be cut off
  - They refresh subscriptions, but unclear how often this has to happen to ensure that the quality will be good
  - (Treat subscriptions as “leases” so that they evaporate if not refreshed... no need to unsubscribe...)
SCRIBE: Failure Management

- Reactive fault tolerance
- Tolerate root and nodes failure
- Tree repair: local impact
  - Fault detection: heartbeat messages
  - Local repair
1500 groups, 100,000 nodes, 1 msg/group

Low delay penalty

Good partitioning and load balancing

- Number of groups hosted per node: 2.4 (mean) 2 (median)

Reasonable link stress:

- Mean msg/link: 2.4 (0.7 for IP)
- Maximum link stress: 4*IP
Topic distribution

![Graph showing topic distribution with labels: Windows Update, Stock Alert, Instant Messaging]
Concern about this data set

- Synthetic, may not be terribly realistic
  - In fact we know that subscription patterns are usually power-law distributions, so that’s reasonable
  - But unlikely that the explanation corresponds to a clean Zipf-like distribution of this nature (indeed, totally implausible)
  - Unfortunately, this sort of issue is common when evaluating very big systems using simulations
  - Alternative is to deploy and evaluate them in use… but only feasible if you own Google-scale resources!
Delay penalty

Mean = 1.66
Median = 1.56
Node stress: 1500 topics

Mean = 6.2
Median = 2

Total number of children table entries
CS5412, Spring 2016
Link stress

Mean = 1.4
Median = 0

Maximum stress

Number of Links

Link stress

CS5412, Spring 2016
T-Man

T-Man

CS5412, Spring 2016
In recent work we’ve been looking at “rigid” overlays

- The data center tracks membership and reports it
- The membership data is used to construct a graph of who will talk to who in each (asynchronous) messaging round

It turns out to be interesting to run gossip on this overlay

- There is a privacy-preserving way to do data mining that uses this approach: the fixed nature of the overlay is useful
- Sensitive data never leaves end-user devices (like smart phones)
- User can query the system but only sees the aggregated data (histograms) not the raw data. Can even tolerate attackers
Things we discovered

- There are really three separate questions here

  1. **What value do overlay networks bring?** For this recent work, they help us efficiently ask questions about private data without leaking it.

  2. **How should we build the overlay itself?** T-Man uses gossip, but this isn’t obligatory. Our new approach uses a special kind of graph.

  3. **What attacks can be launched against the overlay?** This was a weakness of T-Man: it can be tricked into creating a misshapen overlay. Our new approach is attack tolerant.
Overlay networks are a powerful concept that can extend the cloud into the network in a structured way, with many benefits: fault-tolerance, attack tolerance, very efficient structures.

Then we can run tasks like data querying on the overlay, and this can be beneficial.

In the future Internet of Things we will probably see more and more work that use these ideas.