Structured vs. unstructured overlays

Structured Overlays:

- structure $\Longrightarrow$ placement of files is highly deterministic, file insertions and deletions have some overhead
- Fast lookup
- Hash mapping based on a single characteristic (e.g., file name)
- Range queries, keyword queries, attribute queries difficult to support

Unstructured Overlays:

- No structure for overlay $\Longrightarrow$ no structure for data/file placement
- Node join/departures are easy; local overlay simply adjusted
- Only local indexing used
- File search entails high message overhead and high delays
- Complex, keyword, range, attribute queries supported
- Some overlay topologies naturally emerge:
  - Power Law Random Graph (PLRG) where node degrees follow a power law. Here, if the nodes are ranked in terms of degree, then the i^{th} node has c/i^{α} neighbors, where c is a constant.
  - simple random graph: nodes typically have a uniform degree

Unstructured Overlays: Properties

- Semantic indexing possible $\Longrightarrow$ keyword, range, attribute-based queries
- Easily accommodate high churn
- Efficient when data is replicated in network
- Good if user satisfied with "best-effort" search
- Network is not so large as to cause high delays in search

Gnutella features

- A joiner connects to some standard nodes from Gnutella directory
- *Ping* used to discover other hosts; allows new host to announce itself
- *Pong* in response to *Ping*; *Pong* contains IP, port #, max data size for download
- *Query* msgs used for flooding search; contains required parameters
- *QueryHit* are responses. If data is found, this message contains the IP, port #, file size, download rate, etc. Path used is reverse path of *Query*.

Search in Unstructured Overlays

Consider a system with n nodes and m objects. Let q_i be the popularity of object i, as measured by the fraction of all queries that are queries for object i. All objects may be equally popular, or more realistically, a Zipf-like power law distribution of popularity exists. Thus,

$$\sum_{i=1}^{m} q_i = 1 \tag{1}$$

$$\text{Uniform: } q_i = 1/m; \qquad \text{Zipf-like: } q_i \propto i^{-\alpha} \tag{2}$$

Let r_i be the number of replicas of object i, and let p_i be the fraction of all objects that are replicas of i. Three static replication strategies are: uniform, proportional, and square root. Thus,

$$\sum_{i=1}^{m} r_i = R; \qquad p_i = r_i/R \tag{3}$$

$$\text{Uniform: } r_i = R/m; \qquad \text{Proportional: } r_i \propto q_i; \qquad \text{Square-root: } r_i \propto \sqrt{q_i} \tag{4}$$

Under uniform replication, all objects have an equal number of replicas; hence the performance for all query rates is the same. With a uniform query rate, proportional and square-root replication schemes reduce to the uniform replication scheme.

Search in Unstructured Overlays

For an object search, some of the metrics of efficiency:

- probability of success of finding the queried object.
- delay or the number of hops in finding an object.
- the number of messages processed by each node in a search.
- node coverage, the fraction of (distinct) nodes visited
- *message duplication*, which is (#messages - #nodes visited)/#messages
- maximum number of messages at a node
- *recall*, the number of objects found satisfying the desired search criteria. This metric is useful for keyword, inexact, and range queries.
- *message efficiency*, which is the recall per message used

Guided versus Unguided Search. In unguided or blind search, there is no history of earlier searches. In guided search, nodes store some history of past searches to aid future searches. Various mechanisms for caching hints are used. We focus on unguided searches in the context of unstructured overlays.

Search in Unstructured Overlays: Flooding and Random Walk

- Flooding: with checking, with TTL or hop count, expanding ring strategy
- Random Walk: k random walkers, with checking
- Relationships of interest
  - The success rate as a function of the number of message hops, or TTL.
  - The number of messages as a function of the number of message hops, or TTL.
  - The above metrics as the replication ratio and the replication strategy changes.
  - The node coverage, recall, and message efficiency, as a function of the number of hops, or TTL; and of various replication ratios and replication strategies.
- Overall, k-random walk outperforms flooding