Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications

Ion Stoica et al. **TON** 2003

Transactions on **Networking**

Chord

- Breakout discussion:
  - What sorts of **WAN P2P** challenges does Chord have to deal with? And, how does it deal with them?

Chord

- What sorts of **WAN P2P** challenges does Chord have to deal with? And, how does it deal with them?
- **Scalability ~ 10^8 nodes, Privacy, Simplicity <=> No central service (e.g., RSM)**
- **Scale => limit the amount of change that must happen during a node join/leave**
    - Limit number of keys that must be moved (are moved from nearby nodes)
    - Limit the number of routing updates
- Scale => Time complexity of lookup/join must be minimized
    - O(log N) space
- Scale => P2P nodes are sometimes “volunteers”, minimize their resource usage (want to be inclusive of nodes: decrease bar for participation)
    - O(log N) space
- **Scale** => **Diversity of nodes** => highly dynamic behaviours (nodes **Churn**)
    - Focus on join/leave protocol
    - Make node failures and leave the same; efficient
    - Joins of new nodes; must know an existing node to join (bootstrap problem)
- Diversity => Node failures (fail stop)
- **WAN** => Reachability between nodes unclear
    - Routing protocol needs to account for this

Chord

- What sorts of **WAN P2P** challenges does Chord have to deal with? And, how does it deal with them?

- Scale => **Fair** to different nodes => Load balancing: *no central point* of control to distribute resources/nodes
  - Fair: O(log N) storage per node and identical routing state per node
  - Security: more powerful nodes do not carry more of the load
  - *Free riding* is limited
  - Incentive-compatibility as a design goal (not in Chord)
  - Must be completely distributed

Chord and **WAN**

- WAN: wide area networking (versus LAN/data centre)
  - Network heterogeneity (TCP/IP stack mostly resolves this)
- Topology and Reachability
  - **Asymmetry** and **non-transitivity (Chord assumes these away)**
    - Symmetry: A->B => B->A
    - Transitivity : A->B->C ==> A->C
- Geo-distribution of nodes is all over (the world)
  - Orders of magnitude difference in latency (delay) between nodes
  - Routing must account for latency, not just # of hops
  - Solve this in the allocation of nodes to circle
    - Why not: system susceptible to geo-attacks (country, org)
  - Or, solve it in the finger table (Chord solves in the finger table)

Chord

- Iterative versus recursive routing
    - Iterative: bounce from node to source over and o ver again
        - Advantage: Full visibility from the source node
- Recursive: nodes I route to, route on my behalf
    - Advantage: 1/2 the latency on average; no bounce back to source on each hop
    - Disadvantage: failures / black holes (difficulty of diagnosing)

Chord

- Basic setup:
  - Ring with successors pointers
  - Finger tables with exponential hops away
  - Keys assigned to successors on the ring
  - Consistent hashing for balancing nodes and keys
    - Use the same ring for both (key-space)

Chord

- Advanced functionality:
  - Instead of 1 successor => K successors
    - Fault tolerance (ring connectivity)
    - Useful for replication (sets a constant replication factor; easy to find the item). Replicate items close to each other in the key space: efficiency of co-location in key space between replicas.
  - Virtual nodes: further load balancing — make the complexity work in your favour (100 nodes may be not enough X 100 virtual nodes/node => 10,000). Virtual scaling => load balance better
    - Higher storage (linear in virtual nodes)
    - Take care with replication (“many” (in a probabilistic sense) virtual replicas could be on same physical host)
  - Self-stabilization
    - Chord is an example of a Self-* system (self-regulating / self-repairing)

Chord

- *Things the paper doesn't deal with:*
  - Design for incentives to encourage high P2P participation
  - Security
    - Byzantine failures
      - Attack surface is huge: many more types of attacks! Routing, DDoS, introducing asymmetry, sybils
    - P2P-specific: taking advantage of mis-designed incentive in the system (BitTorrent)
  - Bootstrapping new nodes (knowing existing node)
  - Setting RPC/communication timeout
  - Asymmetry and non-transitivity

Chord

- What it can be used to provide:
    - File storage (CFS): store replicas of data at successor nodes
    - Anti-censorship (Tor): fault-tolerant routing can be used to get around routing blocks (route my request to *any* in Chord and try to get to the resource); nodes live all over the world (diversity in routing that is available to a node)
        - By contrast, on Internet: **path cannot be controlled by source** && **hierarchy** means you get consistent path that can be controlled by local authorities.
        - Overlay networks: application-level routing instead of physical route — physical route does not reflect my ultimate destination
            - Only works with recursive routing
- Privacy (Tor, Bitcoin)

Chord and correctness

- Model Chord and prove that it is <u>incorrect</u> (given some “reasonable” discrete time model)
  - *Using Lightweight Modeling To Understand Chord*, by Pamela Zave (related readings posted on schedule)
- There are sequences of joins/leaves/failures/ self-stabilization that lead to broken Chord rings
- Model checking-style methodology

Next: BitTorrent and Bitcoin

- Loosely structured P2P systems
  - File transfer: BitTorrent
  - E-money: BitCoin
- Note: these two papers are short and of questionable quality, *good examples of how **not** to write a paper* (sorry)