Lecture 2: Intro to DS structure and Grapevine

1. Notice: no reviews for today

Overview of distributed systems

a. Desirable Properties
    i. ·Fault-Tolerant: It can recover from component failures without performing incorrect actions.
    ii. Highly Available: It can restore operations, permitting it to resume providing services even when some components have failed.
    iii. Recoverable: Failed components can restart themselves and rejoin the system, after the cause of failure has been repaired.
    iv. Consistent: The system can coordinate actions by multiple components often in the presence of concurrency and failure. This underlies the ability of a distributed system to act like a non-distributed system.
    v. Scalable: It can operate correctly even as some aspect of the system is scaled to a larger size. For example, we might increase the size of the network on which the system is running. This increases the frequency of network outages and could degrade a "non-scalable" system. Similarly, we might increase the number of users or servers, or overall load on the system. In a scalable system, this should not have a significant effect.
    vi. Predictable Performance: The ability to provide desired responsiveness in a timely manner.
    vii. Secure: The system authenticates access to data and services [1]

b. Failure Types
    i. Halting failures: A component simply stops. There is no way to detect the failure except by timeout: it either stops sending "I'm alive" (heartbeat) messages or fails to respond to requests. Your computer freezing is a halting failure.
    ii. Fail-stop: A halting failure with some kind of notification to other components. A network file server telling its clients it is about to go down is a fail-stop.
    iii. Omission failures: Failure to send/receive messages primarily due to lack of buffering space, which causes a message to be discarded with no notification to either the sender or receiver. This can happen when routers become overloaded.
    iv. Network failures: A network link breaks.
    v. Network partition failure: A network fragments into two or more disjoint sub-networks within which messages can be sent, but between which messages are lost. This can occur due to a network failure.
    vi. Timing failures: A temporal property of the system is violated. For example, clocks on different computers which are used to coordinate processes are not synchronized; when a message is delayed longer than a

threshold period, etc.

vii. Byzantine failures: This captures several types of faulty behaviors including data corruption or loss, failures caused by malicious programs, etc. [1]

c.

Eight fallacies: Generally, LAN conditions don't always exist

a. The network is reliable.
    i. What if network is 5 nines reliable – 99.999%. If you send a gigabit of data
    ii. Network can fail for a variety of reasons: backhoes, operators, software failures
    iii. How often is our department cut off from the Internet, or are you at home?
b. Latency is zero.
    i. "*But I think that it's really interesting to see that the end-to-end bandwidth increased by 1468 times within the last 11 years while the latency (the time a single ping takes) has only been improved tenfold. If this wouldn't be enough, there is even a natural cap on latency. The minimum round-trip time between two points of this earth is determined by the maximum speed of information transmission: the speed of light. At roughly 300,000 kilometers per second, it will always take at least 30 milliseconds to send a ping from Europe to the US and back, even if the processing would be done in real time.*"
    ii. *Matters most when waiting for a response (round-tripd elay)*
c. Bandwidth is infinite.
    iii. Getting better, definitely
    iv. Problem comes not from a single client, so much, but from many clients acting simultaneously (e.g. refreshing every X minutes)
    v. Wide-area bandwidth limited by TCP/IP and losses
        1. At 40 msec RTT and 0.1% (1 in 1000 packet) loss, TCP/IP capped at 6.5 Mbps.
        2. To reach 500 Mbps, need 3×10^{-7} error rate
d. The network is secure.
    vi. Example: FTP sends password, username in cleartext
    vii. E.g. MS Windows RPC did not validate format – assume correct. Malformed packet would crash server
    viii. Attacks:
        3. IP injection
        4. Snooping
        5. Denial of service
        6. Dictionary attacks
        7. Malware on client desktops (see Google in China), means firewalls aren't enough

e. Topology doesn't change.
    ix. Machines move, to different networks, different routes
    x. Can't statically say how to route things, where servers are, etc.
    xi. So: use names for indirection (e.g. dns, not ip addresses)
    xii. So: use discovery: ask a service for the best server to use
f. There is one administrator.
    xiii. Cannot change everything at once
    xiv. Cannot change everything at all – e.g. could change server settings but not all client settings
g. Transport cost is zero.
    xv. Network is not free – provided in this department, but for real systems someone must pay for it
h. The network is homogeneous.
    xvi. Latencies, reliability, distances vary
    xvii. E.g.: DSL, dialup, LAN clients, mobile
    xviii. Different speeds, latencies, prices

2.

Grapevine

a. Name server & email system
    i. Name server maps user names to mailboxes and groups to members (other groups or users)
    ii. Email system buffers and delivers messages to inbox, where user downloads them
    iii. NOTE: Same features as MS exchange from 1998
b. What is the scale of this system? An enterprise
    i. 10,000 users
    ii. 30 servers
    iii. A dozen sites
c. QUESTION: What is envisioned environment?
    i. Internet (interconnected LANs) at a single large organization with single management
    ii. Mix of high speed (lan, 56 kb, 8kb links)
d. What are goals?
    i. Scale to many users
    ii. Scale by adding machines, not bigger machines
    iii. User can always send a message
    iv. Tolerate failures of any machine
    v. Decentralized administration
    vi. Large range of user sets – small to very large
e. QUESTION: What problem does this solve?
    i. How to scale a service horizontally (adding more machines) rather than vertically (bigger machines)
    ii. How?

            1. Replicate for reliability
            2. Hierarchically partition data (names, users) to different machines
            3. Avoid global state
            4. Avoid rigid consistency
                a. Write things in one place, replicate later
                b. Don't replicate message contents
        iii. What state doesn't do this?
            1. Set of machines – replicated globally
            2. Originally, members of a distribution list
        iv. As system grows, what makes it scale?
            1. Admin must decide how to partition things, where to add servers, how to incorporate hierarchy into big groups
    f. QUESTION: What do they give up to make this work?
        i. Guarantees: latency can be long
        ii. Consistency: may make an update that is not immediately visible, may have duplicate messages delivered
    g. QUESTION: What are the lessons for distributed systems?
        i. Load can scale beyond capability of a single machine (or a set) leading to congestion collapse
        ii. All state should be partitioned/replicated, avoid global state or having to have all of anything somewhere
        iii. Remote monitoring, local logging helps debug distributed problems
        iv. Makes things fail in an understandable way
            1. Example: file system. Better to have whole directories unavailable than some files in a directory. Lets users work around
            2. Example: reload/stop button on web browser
    h. High-level solutions

        i. Partition work
            1. E.g. hierarchical groups
            2. In distributed systems, some knowledge is global. Handle by:
                a. Keep it small and static (slowly changing), loosely consistent
                b.
        ii. Move data closer to work – not done, but proposed
            1. When sending message to group with users stored on remote computer, could move message to a close message server and do fast, local communication instead of remote communication
            2. For expanding large groups; make it multiple groups on different machines that do local expansion (layer of indirection)
        iii. Caching
            1. Used for repeated access checks – don't need to perform same user/group lookups
        iv. Alternate data structures for more efficient access

1. Store flattened version of nested groups
2. Like index in database
3. Partitioning
    a. Database of users is split into registries, each registry can be stored on different machines
    b. Names partition users based on registry name

v. Spreading load
    1. Put users mailboxes on different machines to avoid hot spots
    2. Put backup mailboxes for users on a single primary on different machine to avoid overload on failure

vi. Reduce service guarantees

vii. Idempotent operations
    1. Allow duplicate messages; avoids cost of expanding groups completely in delivery
    2. Postmark in message allows duplicate detection in mailbox; adding same message multiple times doesn't do anything more
    3. Can avoid expensive sorting, duplicate detection algorithms

viii. Delta encoding
    1. Changes to group memberships sent as deltas (add / remove member) instead of new membership (entire list of members)
    2. Removes need to merge large data objects; just apply change

ix. Expose internals
    1. Naming convention dictates whether a name on an ACL is a user or group; allows for faster lookups because don't need to expand users

x. Input throttling to reduce overload
    1. Servers reject messages when disks are full
    2. Can lead to deadlock

i. How address the fallacies?
    i. Pretty much took into account every one