Section 1: Week 1: Basic Distributed System

Nate Bachmeier

TIM-8120: Distributed Systems

September 22nd, 2019

North Central University

# Distributed System Structure

Contoso Clothing is an international manufacturer and retailer of personal attire. Their manufacturing processes span three continents and need to meet the needs of their several thousand stores. Each location has dozens to hundreds of employees that need to have access to computing environments, point of sales services, and printers.

There are multiple configuration options for the construction of this environment, each with different pros and cons. If they lack the understanding of these trade-offs, then the system will be (1) be too expensive to operate, (2) unable to meet peak loads and (3) unreliable during complex scenarios.

# Command and Control Structured

Traditional systems, such as presented in figure one, might use a static hierarchical structure where the headquarters will replicate policy through a tree structure. Within the tree there each branch represents an aggregation point, such as North America or Europe. There can be child branches to further distribute the load to management points, such as Washington and Spain. These management points will then execute commands on clients (leaf nodes) and collect any local results.

While these trees are good at distributing load across a vast breadth of systems, they introduce several single points of failure (Annadurai & Vijayalakshmi, 2016). To mitigate these risks, administrators implement these branch nodes as complex systems, not individual compute units (Khaneghah & Sharifi, 2013). For instance, the *Washington Management Point* might not be a single server but a load-balanced ring of servers.

By introducing a load-balancer, the administrators are trading availability for additional complexity. Consider the impact of a client sending three requests to the load-balancer, which in turn hands them to three different service instances. These scenarios can lead to (1) out-of-order eventing, (2) partial conversation failure and (3) redundant resource allocations – to name a few challenges.

However, it can simplify other scenarios such as software rejuvenation strategies as there are multiple identical processors within the functional group. Rejuvenation is the operational procedure of recycling private instance state after it has exceeded a threshold (Yang, Min, Yang, & Li, 2013). Since the functional group contains two or more nodes, the rejuvenation can be applied to one node while the other continues to service requests.

Perhaps the message processor leaks memory and becomes unresponsive after the working set exceeds 1GB. In this scenario having an external process (1) monitor the performance metric then (2) cleanly cycle the worker process as it approaches the threshold, would (3) increase the perceived reliability of the message processor.

# Transition Manufacturing Hubs to Pub/Sub