Examining the effects of communication constraints on large scale distributed applications.

Dr. Jacky Mallett

Distributed Applications

What are Distributed Applications?

- Large scale, message based systems – typically scaling up to hundreds of thousands of individual computers
- Real time
- Run on heterogeneous computing platforms, connected by communication networks
- eg. Skype, Web Services, Massively Multiplayer Online Games (MMO's, etc.)

What are MMO's?

Strictly, they are real time distributed *communicating* systems

- Communicating
  - Computation is based on exchange of information between processes – (message passing)
- Real time
  - Behavior of system evolves over time based on previous state, with hard limits on performance in each individual time interval
- Distributed
  - There is no 'single server'
  - The application is spread over a set of connected computers
  - Eve for example, ~150 core servers (proxy + sols) plus client machines
  - ~57,000 machines (PCU max @ Jan 2010)

Special Case of Complex Systems

"A complex system is a network of heterogeneous components that interact nonlinearly…"

- Currently no adequate mathematical description for these systems
- Difficult to predict behaviour over time – provably impossible in many cases
- Combinatorial explosions when modeling at large scale

Many scientific fields are currently blocking on complex systems issues.

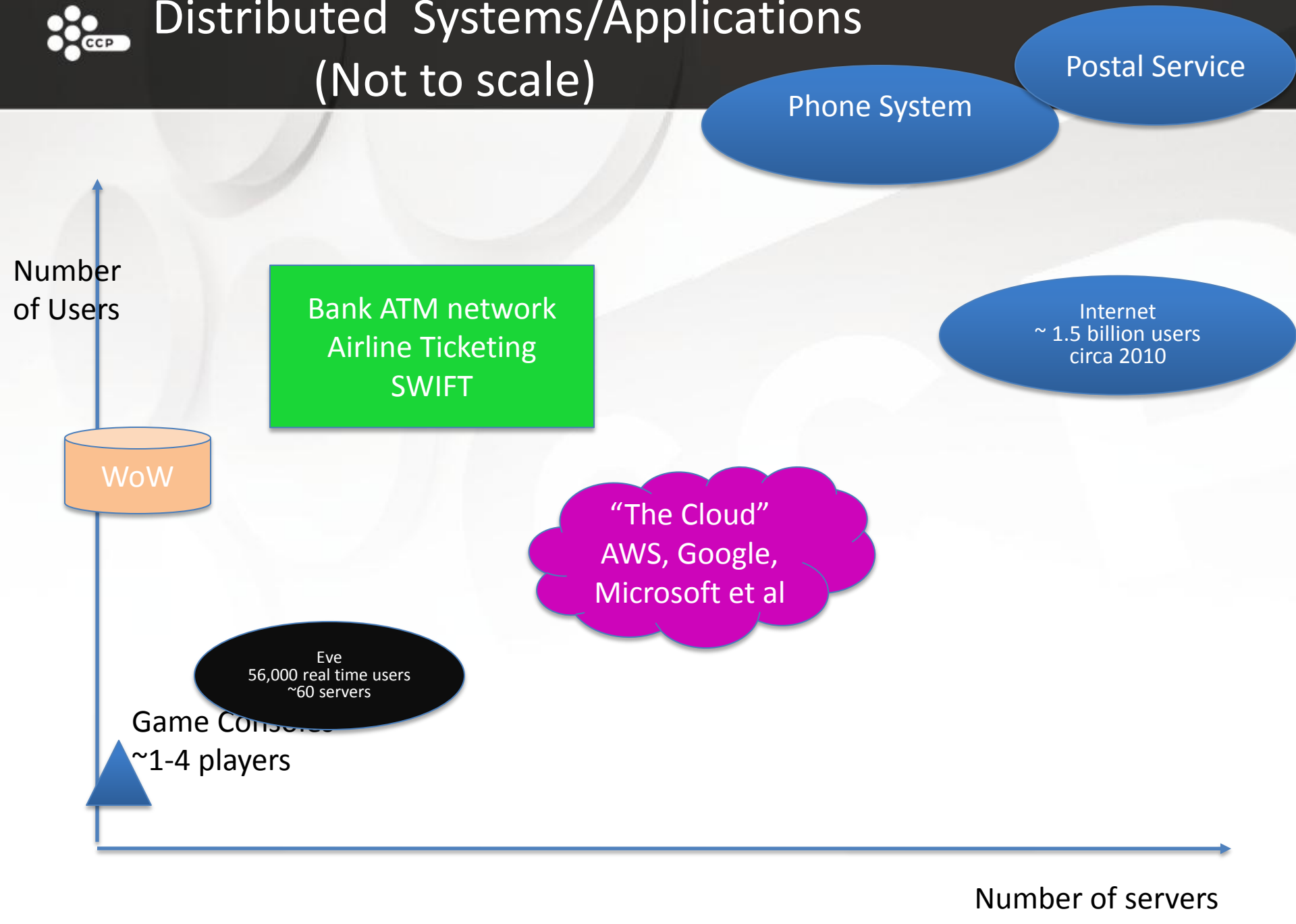
Distributed systems vs. Parallel Algorithms

Important distinction between distributed communicating systems, and parallel algorithms.

As long as there's no communication of data or results between threads, they can be arbitrarily assigned across processors

Communication is the biggest source of problems in these systems

Distributed Systems/Applications (Not to scale)



Distributed Applications

Broadly speaking, we can divide into two groups, based on the underlying network topology:

Hierarchical (Client-Server)	Mesh & Partial Mesh
Telephony	Packet Switched Networking
Web Services	Skype
Bank ATM	P2P (Bittorrent et al)
World of Warcraft MMO	Eve MMO

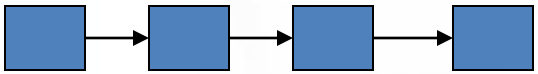
Applications frequently combine both at higher levels of abstraction. E.g. P2P networks are seeded from a single point, and then use mesh based file transfer.

Communicating Groups

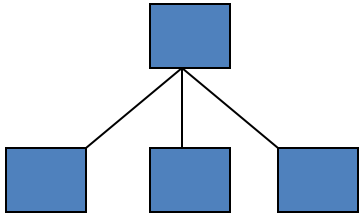
Real time limits on communication occur within a group of communicators:

- Message processing times at nodes (cpu latency)
- Bandwidth between nodes
- Communication speed between nodes (transmission latency)
- Topological Arrangement of the nodes

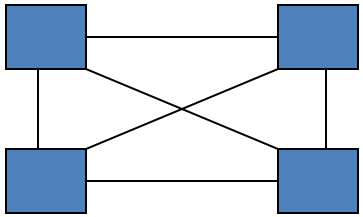
Complex set of interdependent factors



Pipeline



Client-Server

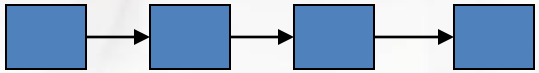


Fully Connected Mesh

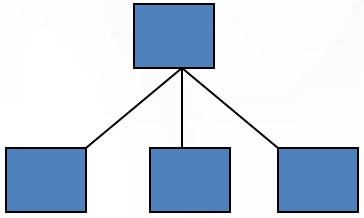
Information Space

Information Space of a Group

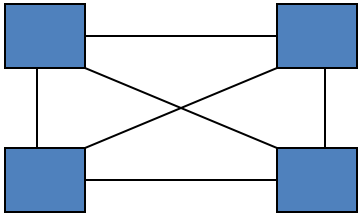
- amount of instantaneous communication that can be performed.
- Assume nodes must send and receive messages asynchronously.



Pipeline



Hierarchical



Fully Connected Mesh

Instantaneous Message Limit

2 messages (each node can either send or receive)

1 message (server bottleneck)

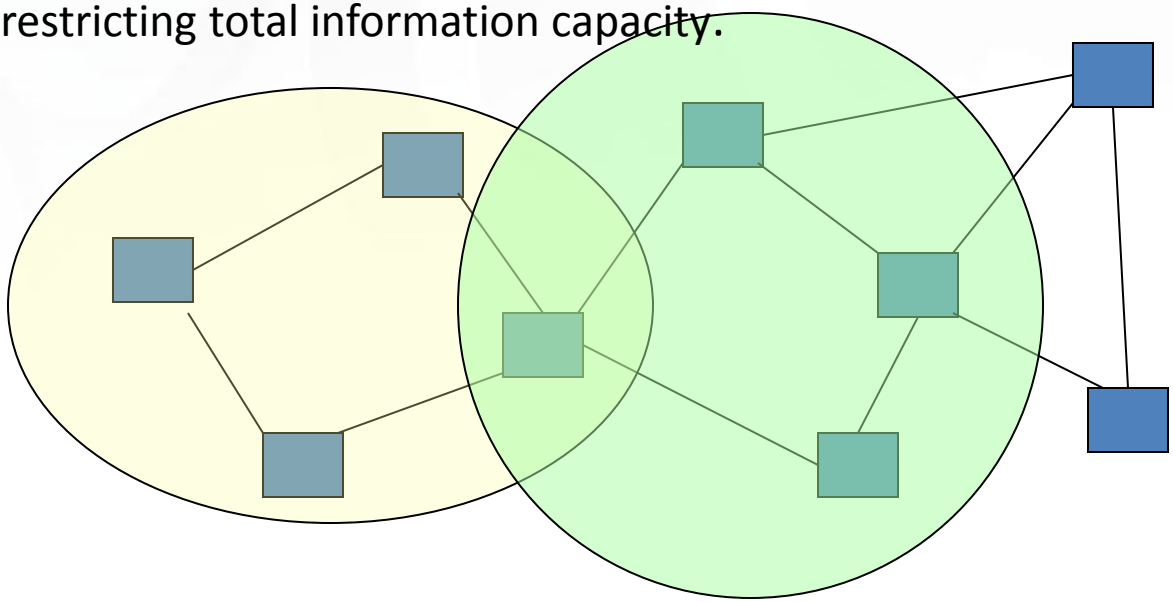
2 messages

Communication based applications are always limited by the singular instance of real time when messages are received

Group Size Limits

The assumption that all members of a group should be able to communicate directly with each other, imposes connectivity limits:

- Defines a group as the *nodes that are accessible within a single hop*.
- At each node there is a limit to the number of direct connections it can support to other nodes
- Below the group size limit, inter-group communication is only constrained by hardware capacity
- Above the group size limit some nodes have to act as relays for other node's messages, restricting total information capacity.



Information Capacity.

Define Information capacity of a network or group as:

Limit on the amount of instantaneous information (messages) that can be transmitted by a network.

Strictly Hierarchical: Link capacity of server, L

Partial Mesh Network: L √N*

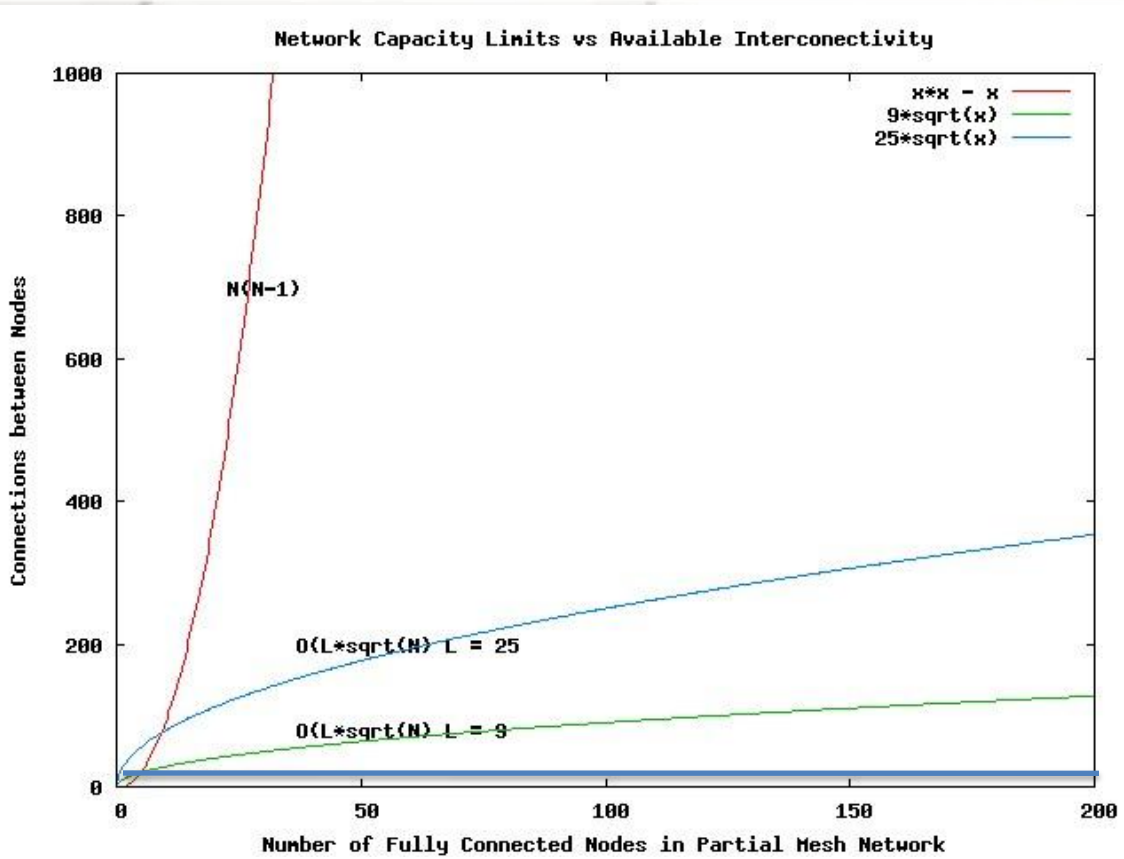
N Number of nodes in network.

L Link capacity of a node (number of links it can support to other nodes for the required message load)

*Gupta P. and P. R. Kumar. 2000. The Capacity of Wireless Networks

Scaglione, Anna and Sergio Servetto. 2002. On the Interdependence of Routing and Data Compression in Multi-Hop Sensor Networks

Information Space for Partial Mesh Topologies

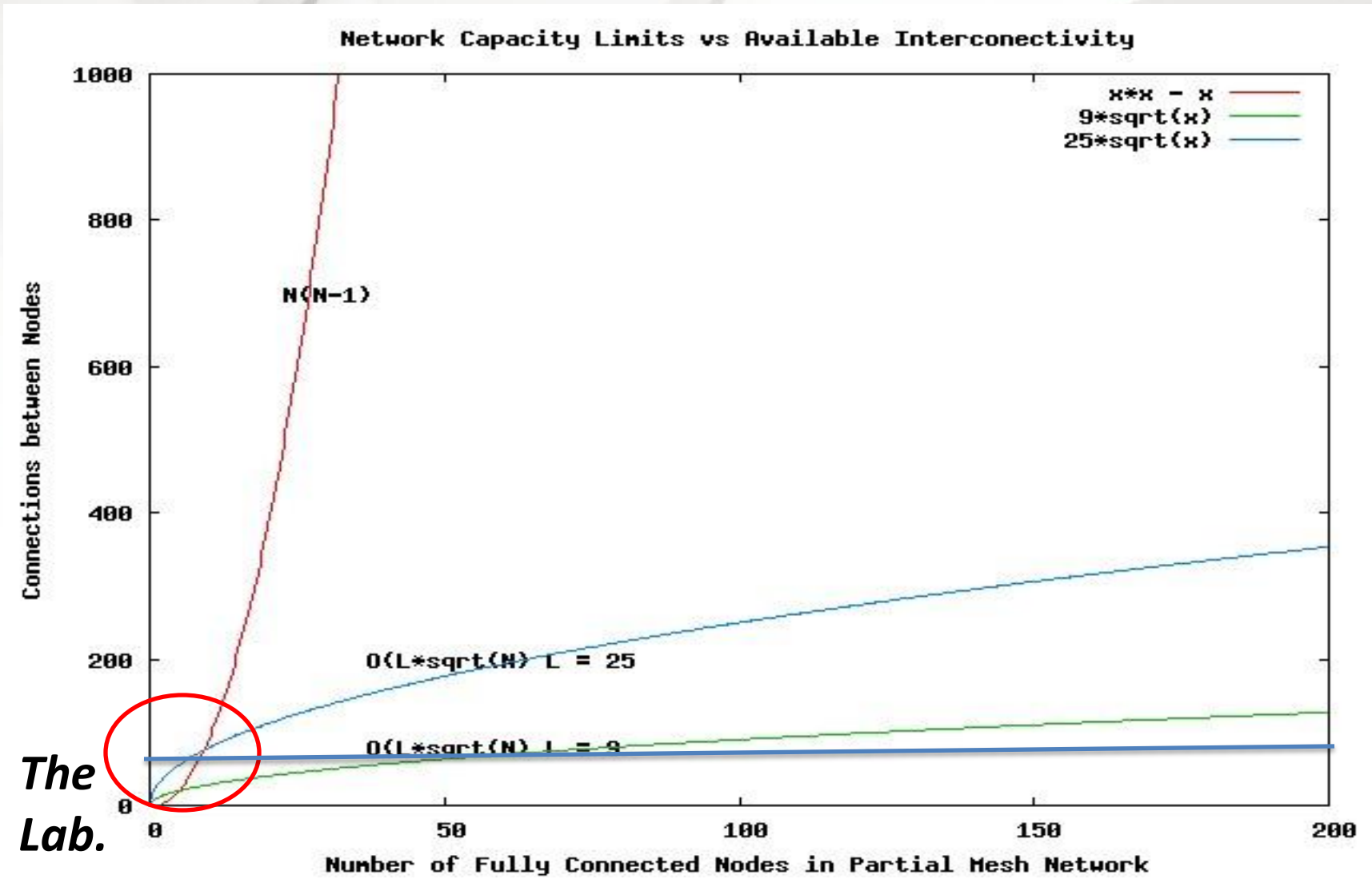


If all nodes can autonomously originate messages, required network capacity is O(N(N-1)).

However, maximum capacity for a partial mesh network is O(LVN)

Beyond the group size/single hop connectivity limit, Information Space does not scale with additional Nodes even with a mesh based architecture.

for small N, there is a special place where everything works.



The Lab.

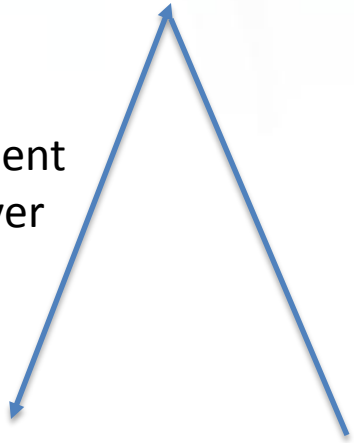
Latency

- Communication and Computational Latency acts as an “invisible” force in these systems
- Communication Latency – time taken to send and receive a message
- Computational Latency – time taken to perform processing consequent on message.
- Message rate at server ≈ application RTT , client -> server

Server CPU Processing

With the assumption that clients send messages on receipt of server response

RTT Client -> Server

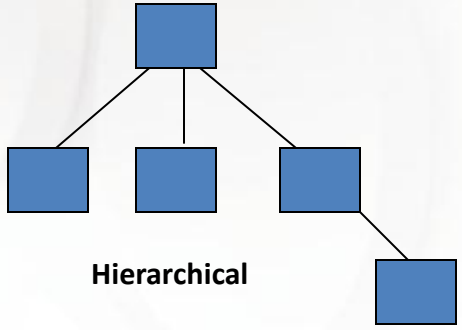


RTT(Message Latency)	Server CPU	# Clients
1s	10ms	100
100ms	10ms	10
10ms	10ms	1

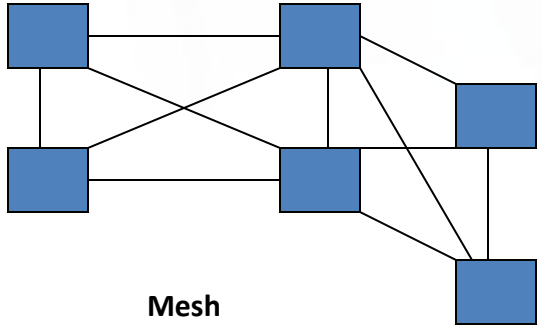
Information versus Control

Controlling a group:

- Requires agreement on task, and communication to all involved nodes



Hierarchical



Mesh

Hierarchical Networks are optimised for rapid control from a central point.

Mesh networks are optimised for maximum exchange of information.

For large scale distributed applications, above the group size, the control path is inevitably faster than the return information flow.

How long does the group have to make the decision?

Advantages & Disadvantages

Hierarchical	Mesh
Provides single point of control/Synchronization	No single point for control
Information Capacity is O(Server)	Information capacity is O(LVN)
Single point of failure	Robust to multiple failures
Excess load is inherently throttled	No intrinsic throttling – the end points can always overload the network
Optimal for control purposes	*Optimal for Information sharing purposes*

Latency also plays a part:

- As latency increases, relative advantages of Mesh Information Sharing drop
- Large Scale, long latency applications with control requirements may not be able to wait for information from all nodes before acting

As do Fisher Consensus Requirements

- Synchronization needs dictate a hierarchical topology in many applications

Design Questions

- What are the advantages and disadvantages of the two topology groups?
- Why are particular topologies used for different applications?
- What are the limits of particular design approaches?

MMO System Scaling Issues

- Fast real time interaction between players
  - Eg. fighting
  - Scales $O(N^2)$ or worse – e.g. autonomous missiles
- Slow real time interaction
  - Eg. Chat, visual contact, status updates
- Transfers between servers
  - State propagation (Consensus Issues)
  - Latency change with inter-server communication
- Inventory
  - O(No. of Players * amount of stuff each one can have)
  - In Eve, corporations can also own stuff
  - Creates database scaling issues, and also some real time game issues if large moves are performed
  - Eve players are packrats

Game Design Scaling Issues

- Room design
  - Scaling issue when producing MMO games
  - Sharding allows a lot of replicated content, content of game can be developed slower
    - Or allow players to generate/design own content
    - Player owned stations in Eve, Second Life
  - Topological organisation of rooms can create scaling issues at different levels of the system from emergent behaviour
    - e.g. Jita
- Game Balance
  - Presents as various scaling and complexity issues and should probably be its own field of research

- Distributed computing issues apply at all levels of abstraction
  - Inter-process communication ≈ Inter-player communication
  - Modify for Human layer distributed issues – considerably slower, less reliable, and much lower scaling capability (group size)

Design Intuition

- Complex problems are solved by breaking them down into smaller, simpler problems
- Break the complex system down into elements whose behaviour can be modeled and at least estimated. (Back of the envelope is fine)
- Ideally, break it down into elements that do as little as possible communication with each other.

- Topology – arrangement of communication between nodes
- Information capacity – how many messages are required/time period
- Transmission Latency – how long does it take to receive a message
- Computational Latency – how long does it take to process a message
- Group size – function of latency, topology, and information capacity

Designing Large Scale Distributed Applications

- Communication Limits are unavoidable once an application goes beyond its individual group size limit
  - Change the problem so that it is below the group size limit (or improve the hardware)
- Eve Online
  - Physics simulation is performed on client and servers
  - Communication efficient < 1 message/s
  - Direct tradeoff of CPU on server vs message rate to client

- Break problem into distributed groups
  - Now have to deal with topology tradeoffs
  - As much as possible, isolate shared information to sub-groups
  - Solve problems as close as possible to the source of information that creates or resolves them.

Recognise and Design for Limits

- Control requirements impose a hierarchical topology
- Information sharing is maximised with mesh topologies
- Some application spaces are unobtainable
  - The capacity for universal access to information scales badly with the number of nodes in the system
  - Broadcast technology has the lowest possible information space
- In the limit distributed applications are limited by general communication needs not local CPU
  - Can frequently tradeoff CPU vs communication to increase capacity
  - Pre-distribute information and access using simple messages cf. American Football Playbook (tradeoff of memory vs communication)

Challenges

- There are usually multiple ways to solve the majority of distributed systems problems
  - Each solution will have tradeoffs, and typically some of them will be mutually orthogonal
- They can often also be solved and created at each layer of abstraction
- Each solution will have and advantages and disadvantages
- Beware small world solutions if you want anything to scale
- Be honest about what costs are being shifted onto other layers/(teams)
- Avoid complex solutions – it's hard enough getting the simple ones to work.