Distributed Systems

Basic Algorithms

Rik Sarkar

University of Edinburgh

2016/2017
Distributed Computation

- How to send messages to all nodes efficiently
- How to compute sums of values at all nodes efficiently

- Network as a graph
- Broadcasting messages
- Computing sums in a tree
- Computing trees in a network
- Communication complexity
Network as a graph

- Network is a graph: $G = (V,E)$
- Each vertex/node is a computer/process
- Each edge is communication link between 2 nodes
- Every node has a Unique identifier known to itself.
  - Often used 1, 2, 3, ..., $n$
- Every node knows its neighbors – the nodes it can reach directly without needing other nodes to route
  - Edges incident on the vertex
  - For example, in LAN or WLAN, through listening to the broadcast medium
  - Or by explicitly asking: Everyone that receives this message, please report back
- But a node *does not* know the rest of the network
Example: Unit disk graphs

- Suppose all nodes are wireless
- Each can communicate with nodes within distance $r$.
- Say, $r = 1$

- UDG is a model
- Not perfect

- In general, networks can be any graph
Directed graphs

• When A can send message to B, but B cannot send message to A

• For example, in wireless transmission, if B is in A’s range, but A is not in B’s range
Directed graphs

• When A can send message to B, but B cannot send message to A
• Or if protocol or technology limitations prevent B from communicating with A
Directed graphs

• Protocols more complex
• Needs more messages
Network as a graph

• Distance/cost between nodes p and q in the network
  – Number of edges on the shortest path between p and q (when all edges are same: unweighted)

• Sometimes, edges can be weighted
  – Each edge e = (a,b) has a weight \( w(e) \)
  – \( w(e) \) is the cost of using the communication link e (may be length e)
  – Distance/cost between p and q is total weight of edges on the path from p to q with least weight
Network as a graph

• Diameter
  – The maximum distance between 2 nodes in the network

• Radius
  – Half the diameter

• Spanning tree of a graph:
  – A subgraph which is a tree, and reaches all nodes of the graph
  – If network has n nodes
    • How many edges does a spanning tree have?
Computing sums in a tree

• Suppose root wants to know sum of values at all nodes
Computing sums in a tree

• Suppose root wants to know sum of values at all nodes
• It sends “compute” message to all children
• They forward the message to all their children (unless it is a leaf node)
• The values move upward from leaves
• Each node adds values from all children and its own value
• Sends it to its parent
Computing sums in a tree

• What can you compute other than sums?

• How many messages does it take?

• How much time does it take?
Communication complexity

• Used to represent communication cost for general scenarios
• Called Communication Complexity or Asymptotic communication complexity

• Use big oh notation: $O$
Big oh – upper bounds

• For a system of n nodes,

• Communication complexity $c(n)$ is $O(f(n))$ means:
  – There are constants $a$ and $N$, such that:
  – For $n > N$: $c(n) < a \cdot f(n)$

Allowing some initial irregularity, ‘$c(n)$’ is not bigger than a constant times ‘$f(n)$’

In the long run, $c(n)$ does not grow faster than $f(n)$
Examples

- $3n = O(?)$
- $1000n = O(?)$
- $n^2/5 = O(?)$
- $10 \log n = O(?)$
- $2n^3 + n + \log n + 200 = O(?)$
- $15 = O(?)$
Examples

• $3n = O(n)$
• $1000n = O(n)$
• $n^2/5 = O(n^2)$
• $10\log n = O(\log n)$
• $2n^3+n+\log n+200 = O(n^3)$
• 15 or any other constant $= O(1)$
Example 1

- ‘Star’ network
- Computing sum of all values
- Communication complexity: $O(n)$
Example 2a

- ‘Chain’ topology network
- Simple protocol where everyone sends value to server
- Communication complexity:?
Example 2a

- ‘Chain’ topology network
- Simple protocol where everyone sends value to server
- Communication complexity: $1+2+\ldots+n = O(n^2)$
Example 2b

• ‘Chain’ network
• Protocol where each node waits for sum of previous values and sends
• Communication complexity: $1+1+\ldots+1 = O(n)$
Computing sums in a tree

• How many messages does it take?

• How much time does it take?
Global Message broadcast

• Message must reach *all nodes in the network*
  – Different from broadcast transmission in LAN
  – All nodes in a large network cannot be reached with single transmission
Global Message broadcast

• Message must reach *all nodes in the network*
  – Different from broadcast transmission in LAN
  – All nodes in a large network cannot be reached with single transmissions

Distributed Systems, Edinburgh, 2016
Flooding for Broadcast

• The source sends a *Flood* message to all neighbors

• The message has
  – Type *Flood*
  – *Unique id*: *(source id, message seq)*
  – *Data*
Flooding for Broadcast

• The source sends a *Flood* message, with a unique message id to all neighbors

• Every node $p$ that receives a flood message $m$, does the following:
  - *If* $m.id$ *was seen before, discard* $m$
  - *Otherwise, Add* $m.id$ *to list of previously seen messages and send* $m$ *to all neighbors of* $p$
Flooding for broadcast

• Storage
  – Each node needs to store a list of flood ids seen before
  – If a protocol requires $x$ floods, then each node must store $x$ ids
    • (there is a way to reduce this. Think!)
Assumptions

- We are assuming:
  - Nodes are working in synchronous *communication rounds* (e.g. *transmissions occur in intervals of 1 second exactly*)
  - Messages from all neighbors arrive at the same time, and processed together
  - In each round, each node can successfully send 1 message to each neighbor
  - Any necessary computation can be completed before the next round
Communication complexity

• The message/communication complexity is:
Communication complexity

• The message/communication complexity is:
  – $O(|E|)$
Communication complexity

• The message/communication complexity is:
  – $O(|E|)$
  – Worst case: $O(n^2)$
Reducing Communication complexity (slightly)

- Node p need not send message m to any node from which it has already received m
  - Needs to keep track of which nodes have sent the message
  - Saves some messages
  - Does not change asymptotic complexity
Time complexity

• The number of rounds needed to reach all nodes: \textit{diameter of }G
Computing Tree from a network

• BFS tree
  – The Breadth first search tree
  – With a specified root node
BFS Tree

• Breadth first search tree
  – Every node has a parent pointer
  – And zero or more child pointers

  – BFS Tree construction algorithm sets these pointers
BFS Tree Construction algorithm

• Breadth first search tree
  – The root(source) node decides to construct a tree
  – Uses flooding to construct a tree
  – Every node p on getting the message forwards to all neighbors
  – Additionally, every node p stores parent pointer: node from which it first received the message
    • If multiple neighbors had first sent p the message in the same round, choose parent arbitrarily. E.g. node with smallest id
  – p informs its parent of the selection
    • Parent creates a child pointer to p
BFS Tree

• Property: BFS tree is a shortest path tree
  – For source s and any node p
  – The shortest path between s and p is contained in the BFS tree
Time & message complexity

- Asymptotically Same as Flooding
Tree based broadcast

• Send message to all nodes using tree
  – BFS tree is a *spanning* tree: connects all nodes

• Flooding on the tree

• Receive message from parent, send to children
Tree based broadcast

• Simpler than flooding: send message to all children

• Communication: Number of edges in spanning tree: n-1
Aggregation: Find the sum of values at all nodes

- With BFS tree

- Start from *leaf* nodes
  - Nodes without children
  - Send the value to parent

- Every other node:
  - Wait for all children to report
  - Sum values from children + own value
  - Send to parent
Aggregation

• Without the tree
• Flood from all nodes:
  – $O(|E|)$ cost per node
  – $O(n*|E|)$ total cost: expensive
  – Each node needs to store flood ids from n nodes
    • Requires $\Omega(n)$ storage at each node
  – Good fault tolerance
    • If a few nodes fail during operation, all the rest still get some value
Aggregation

• With Tree

• Also called Convergecast
Aggregation

- With Tree

- Once tree is built, any node can use for broadcast
  - Just flood on the tree

- Any node can use for convergecast
  - First flood a message on the tree requesting data
  - Nodes store parent pointer
  - Then receive data

- What is the drawback of tree based aggregation?
Aggregation

• With Tree

• Once tree is built, any node can use for broadcast
  – Just flood on the tree

• Any node can use for convergecast
  – First flood a message on the tree requesting data
  – Nodes store parent pointer
  – Then receive data

• Fault tolerance not very good
  – If a node fails, the messages in its subtree will be lost
  – Will need to rebuild the tree for future operations
BFS trees can be used for routing

• From each node, create a separate BFS tree
• Each node stores a parent pointer corresponding to each BFS tree
• Acts as routing table
BFS trees can be used for routing

• From each node, create a separate BFS tree
• Each node stores a parent pointer corresponding to each BFS tree
• Acts as routing table
• $O(n*|E|)$ message complexity in computing routing table
Observation on complexity

• Suppose $c(n) = n$
  – Then $c(n)$ is $O(n)$ and also $O(n^2)$
  – Although, when we ask for the complexity, we are looking for the tightest possible bound, which is $O(n)$
Big $\Omega$ – lower bounds

- For a system of $n$ nodes,
- Communication complexity $c(n)$ is $\Omega(f(n))$ means:
  - There are constants $a$ and $N$, such that:
  - For $n>N$: $b*f(n) < c(n)$

Allowing some initial irregularity, ‘$c(n)$’ is not smaller than a constant times ‘$f(n)$’

In the long run, $f(n)$ does not grow faster than $c(n)$

Distributed Systems, Edinburgh, 2016
Big $\Theta$ – tight bounds: both $O$ and $\Omega$

- For a system of $n$ nodes,
- Communication complexity $c(n)$ is $\Theta(f(n))$ means:
  - There are constants $a, b$ and $N$, such that:
  - For $n > N$:
    $$b * f(n) < c(n) < a * f(n)$$

Allowing some initial irregularity, $c(n)$ and $f(n)$ are within constant factors of each other. In the long run, $c(n)$ grows at the same rate as $f(n)$, upto constant factors.

Distributed Systems, Edinburgh, 2016
Bit complexity of communication

- We have assumed that each communication is 1 message, and we counted the messages.
- Sometimes, communication is evaluated by bit complexity – the number of bits communicated.
- This is different from message complexity because a message may have number of bits that depend on \( n \) or \(|E|\).
- For example, node ids in message have size \( \Theta(\log n) \).

- In practice this is may not be critical since \( \log n \) is much smaller than packet sizes, so it does not change the number of packets communicated.
- But depending on what other data the algorithm is communicating, sizes of messages may matter.
Size of ids

• In a network of $n$ nodes
• Each node id needs $\Theta(\log n)$ (that is, both $O(\log n)$ and $\Omega(\log n)$) bits for storage
  – The binary representation of $n$ needs $\log_2 n$ bits

• $\Omega$ – since we need at least this many bits
  – May vary by constant factors depending on base of logarithm
Computing Trees:

- What if the edges have weights?
Aggregation using Trees:

• What if the edges have weights?
• The cost may not be $O(n)$ since weights can be higher

• How to get the best performance?
Minimum spanning tree is

• A spanning tree (reaches all nodes)
• With minimum possible total weight

• How can we compute a minimum spanning tree efficiently in a distributed system?
• (remember, a node knows only its neighbors and edge weights)