Distributed Routing

EECS 228

Abhay Parekh

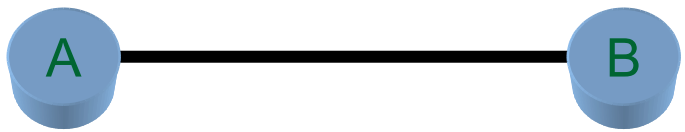
parekh@eecs.berkeley.edu

The Network is a Distributed System

- Nodes are local processors
- Messages are exchanged over various kinds of links
- Nodes contain sensors which sense local changes
- Nodes control the network jointly
  - Method for doing this is a distributed algorithm
- Time taken to solve the problem has two components:
  - Computation time taken for local processing
  - Communication time for messages to be received over the links

Consensus Problem

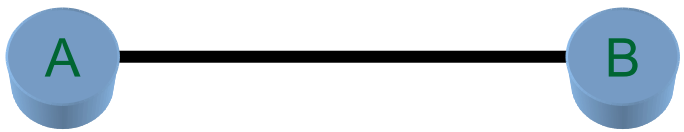
- A and B in a connection over an unreliable link
- They both want to terminate the connection only if they are certain that no more packets will arrive from the other user



- A won't terminate unless it knows that B knows it is about to terminate.
- B won't terminate unless it knows that A knows it is about to terminate

Consensus Problem

- Suppose B tells A it can terminate and A receives this message, say M
- A can terminate, but B will never know if A actually received M and so it can't terminate



- A sends ACK(M) to B, but then A needs to makes sure that B received this message, so it must wait for ACK(ACK(M))…
- A never terminates.
- In fact, NO protocol exists to solve this problem!
- Worth convincing yourself of this fact.

Synchronous v/s Asynchronous Algorithms

- Synchronous algorithms can be described in terms of global iterations. The time taken for a given iteration is the time taken for the slowest processor to complete that iteration: *time driven*
  - E.g. slotted systems like TDM or SONNET allow for synchronous algorithms
- Asynchronous algorithms execute at a processor based on received messages and internal state: *event driven*
  - *E.g.* IP protocols which must run over heterogeneous systems are asynchronous

Links are inherently unreliable

- Error correction
  - Assume that errors can “eventually” corrected
    - Otherwise must assume periodic updates
- Propagation Delay
  - Fixed
  - Variable but no more than d
  - Variable with no upper bound
- Other components of delay
  - Queueing Delay
  - Transmission Delay
- Packet order
  - FIFO
  - Can be delivered in arbitrary order

Soft State

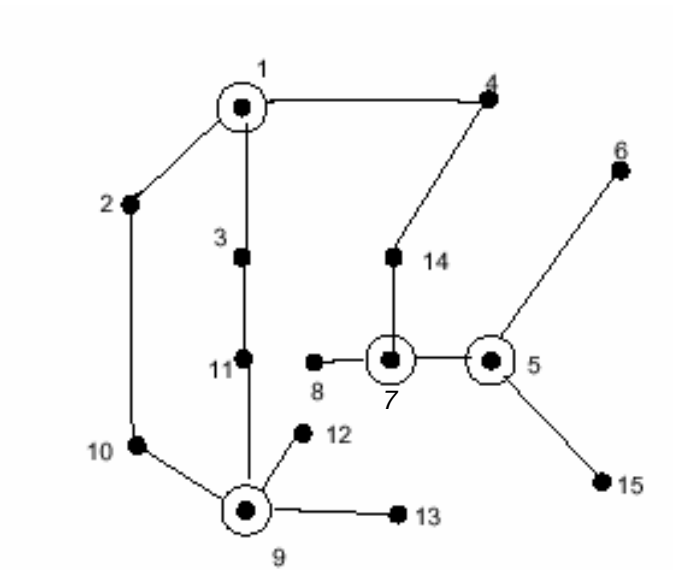
- State with Time-Out
- Example: A host joins a group by sending a “join” message to a “host manager”. The manager adds the host to the group for the next T seconds. If the host wants to stay in the group it must send a refresh message within T seconds to the manager. Otherwise it is dropped.
- Advantage: Manager robust to host failure
- Disadvantage: Too many messages
- Most internet protocols use this way of communicating
- Trades of simplicity of correctness with complexity of communication

Solving Global Problems in a Distributed Setting

- Examples:
  - Minimum Spanning Tree
  - Shortest Path
  - Leader Election
  - Topology Broadcast
- Much easier to think in terms of centralized algorithms
- How does one “convert” to a distributed setting?

Example: Electing Leaders

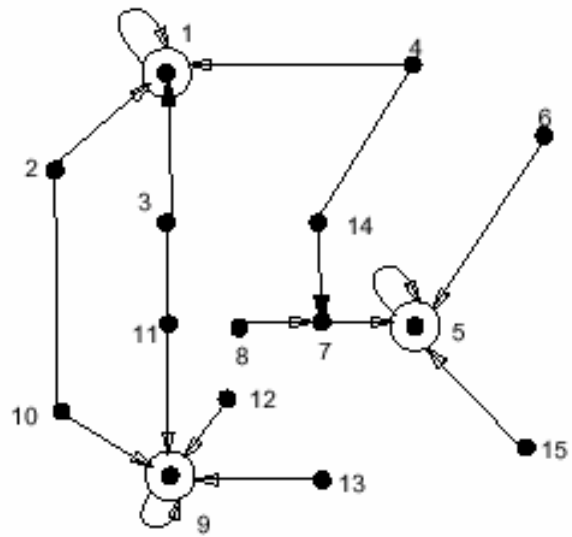
- Global Problem
  - Given an undirected graph with nodes, find a small set of nodes such that every node not in the set has a neighbor in the set. (Dominating Set)
- Finding the smallest set is NP-hard so use a simple greedy algorithm which does the best you can hope for…
- What if topology were changing and decisions need to be made based on local topology information?



Order: 9, 1, 5, 7

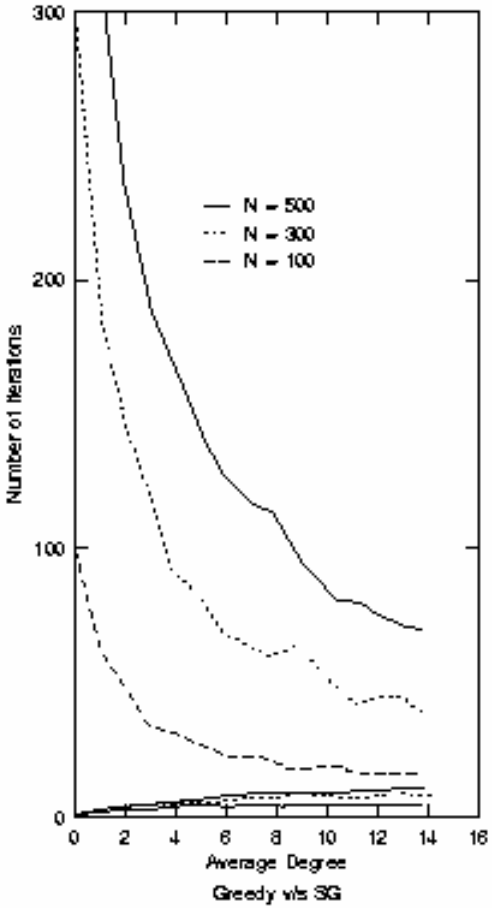
Synchronous Distributed Version

- What if the nodes only know their topologies two hops out?
  1. Find most connected neighbor (vote) and broadcast the vote (terminate if all dominated)
  2. Any node unanimously elected by undominated neighbors joins the dominating set
  3. Election results broadcast
  4. Back to Step 1



Iteration 1: 1,5,9
Iteration 2: 7

Greedy saves time over *SG*



Worst Case: *Greedy* as slow as *SG*
Random Graph Models:

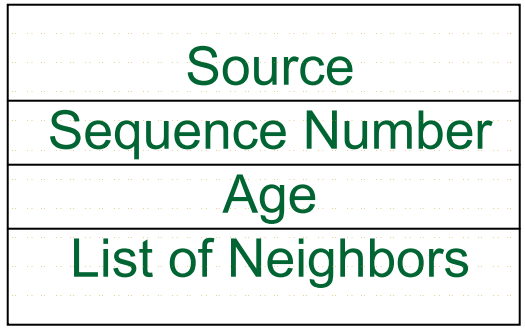
1. N,p: for each node pair, an edge exists with prob p.
2. N,L,r: generate N points uniformly in $(r, L-r) \times (r, L-r)$.
   Edge (i,j) iff dist$(i,j) \leq r$.

IBM

Routing Protocols

- **Addressing:** Uniquely identify the nodes
  - host IP address, group address, attributes
  - set is dynamic!
- **Topology Update:** Characterize and maintain connectivity
  - Discover topology
  - Measure “distance” (one or more metric)
  - Dynamically provision (on slower timescale)
- **Resource Discovery:** Find node identifiers of the destination set
- **Route Computation:** Pick the tree (path)
  - Kind of path: Multicast, Unicast
  - Global or Distributed Algorithm
  - Policy
  - Hierarchy
- **Switching:** Forward the packets at each node

Flooding Link State Information



- LSPs arrive and wait in buffers to be “accepted”
- If node j receives a LSP from node k it compares the sequence numbers. If this is the most recent one from k, send to N(j)-{k}.
- Age starts out at 7. At any router, value is decremented every 8 seconds. At 0 discard.
- Looks reasonable, but crashed the ARPANET
  - See “Interconnections” book by Radia Perlman

Pathological Behavior

- Sequence numbers from some router, s, wrapped around
  - A < B < C < A
- Router, t, has a buffer with LSPs from s of all three values in order: A, B, C
  - Store and flood A
  - Replace A with B and flood B
  - Replace B with C and flood C
- Router u receives the LSPs in order ABC and goes through the same cycle and sends to v
- The entire Arpanet was sending these LSPs and crashed
- LSPs did not wait in buffers long enough to age…

Improved Algorithm: More Complicated!

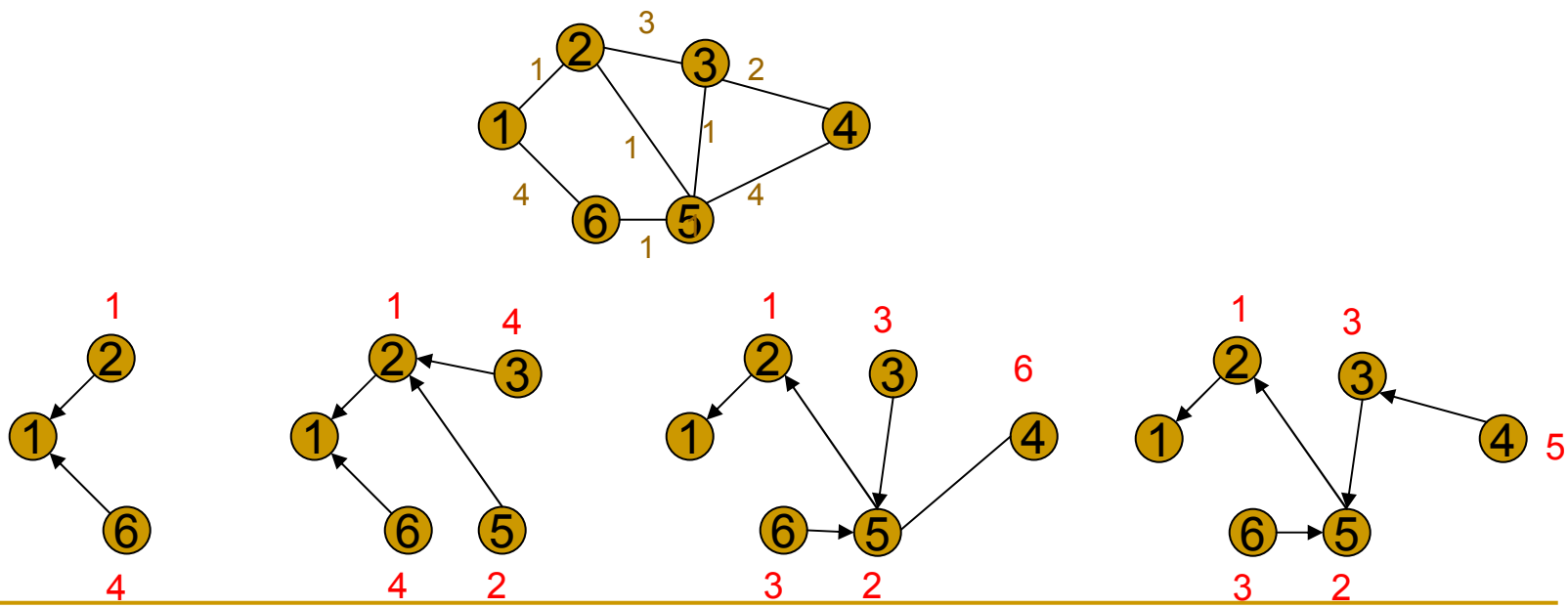
- Don't be in a hurry to flood
- Acknowledge each LSP
- For each LSP, have two flags for each neighbor, i.e. 2|N( )| flags
  - One for Sending and one for ACKing
- When an LSP is received set the appropriate flags
- When bandwidth is available RR the LSP entries to be "fair" and upon seeing the first Send or ACK set flag transmit the LSP or ACK, as appropriate.
- Age as before but
  - Age 0 LSPs are not accepted unless there is another LSP from the same source already in the database
  - Accepted Age 0 LSPs are ACKed, and transmitted. Only deleted when ACKed by all neighbors

Other issues

- What happens if some routers are much faster at transmitting LSPs?
- What happens when a partitioned network is reconstituted?
- What about security?
- Etc., etc.
- Many lines of code

Bellman-Ford Shortest Path

- Shortest walk of ≤ h hops from i to 1 is $D^h(i)$. Stipulate $D^h(i)$ =0 for all h.
- Suppose the first hop in a h+1 shortest hop walk from i is at node j.
  - Then $D^{h+1}(i) = D^h(j) + d_{ij} = \min_k [D^h(k) + d_{ik}]$
- If all link lengths >0, then we get paths not just walks
- Algorithm completes when hop distances do not change any more

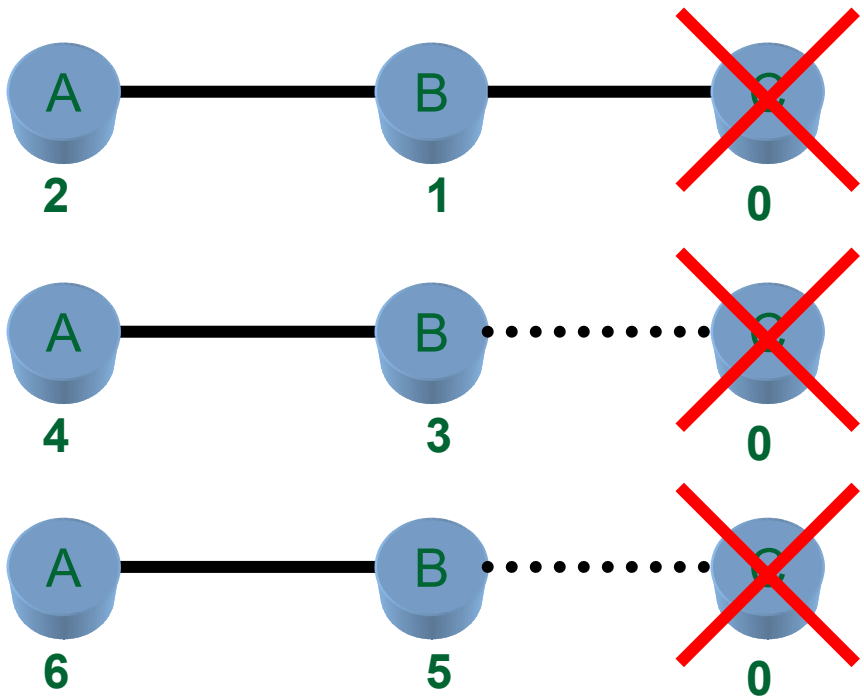


Distributing Bellman Ford: Synchronous

- Each node just knows the costs of the links to its neighbors
- Iteration h+1
  - $D^{h+1}(i) = \min_{k \, \varepsilon \, N(i)} [D^h(k) + d_{ik}]$
  - Broadcast new estimates
- Easy! But…
  - How to get all the nodes to start?
  - What if the a link changes? How to abort?

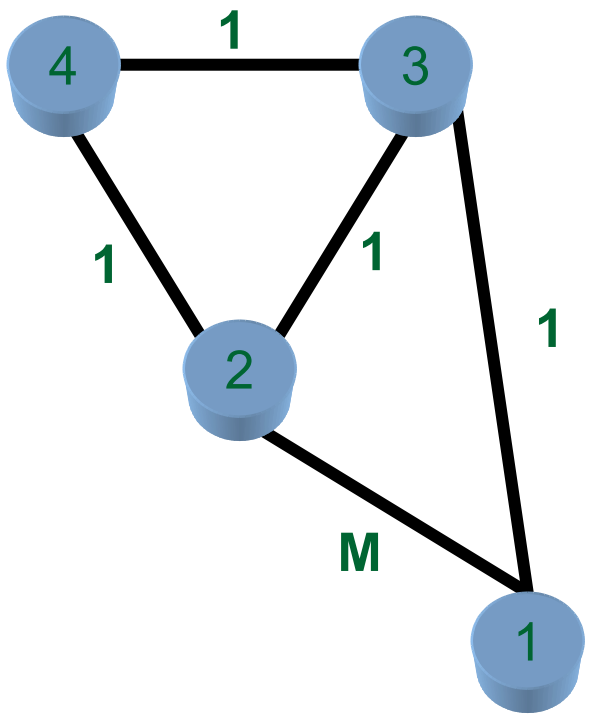
Counting to Infinity

All links cost 1



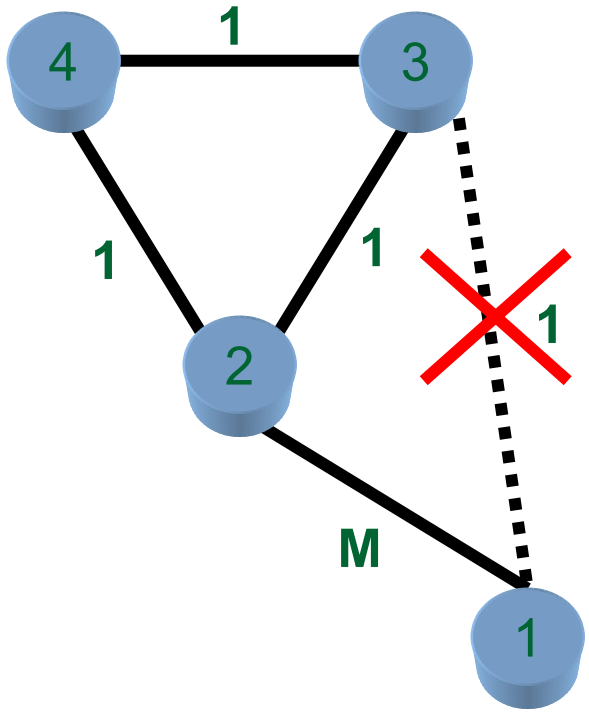
Ping-Pong to Eternity

Bad News Travels Slowly…



D(2)=2, D(3)=1, D(4)=3

Bad News Travels Slowly…



D(2)=2, D(3)=1, D(4)=3

Node 2 takes about M Iterations to figure out that D(2)=L

Initial Conditions and BF Convergence

Let m_i be the smallest number of edges in a shortest path from i to 1

$$m^* = \max_{1,2,\ldots,n} m_i \leq n-1$$

Also,

$$\beta = \max_{1,2,\ldots,n} (x_i^* - x_i^0)$$

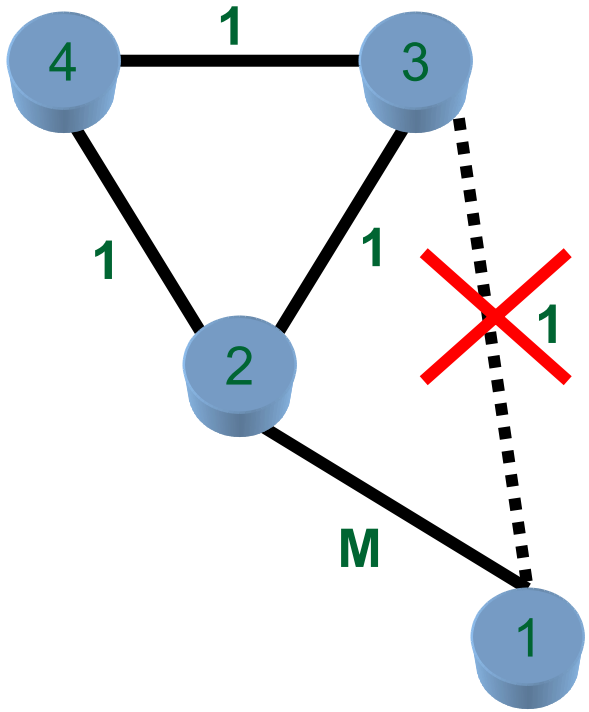
and

$$L = \min_{\text{All cycles}} \frac{\text{Length of Cycle}}{\text{Number of arcs on cycle.}}$$

Then Bellman Ford terminates after T+1 iterations where

$$T = \begin{cases} m^* & \text{if } \beta \leq 0 \\ n-1 & \text{if } \beta > 0 \text{ and the graph is acyclic} \\ n-2+\lceil \beta/L \rceil & \text{if } \beta > 0 \text{ and the graph has cycles} \end{cases}$$

Bad News Travels Slowly…



D(2)=2, D(3)=1, D(4)=3

Node 2 takes about M Iterations to figure out that D(2)=M

β = M – 2
L = 1
Terminates in 4-1+M-2= M+1 iterations

Asynchronous Bellman Ford

- Surprisingly simple…
  - Iterate $D(i) = \min_{k \varepsilon N(i)} [D(k) + d_{ik}]$
  - Broadcast D(i) to N(i)
  - Use last received values of D() and d
- In general, nodes are using different and possibly inconsistent estimates
- If no link changes after some time t, the algorithm will eventually converge to the shortest path
- No synchronization required at all…

The nature of asynchronous distributed protocols

- Generally non-intuitive
- Limited theory to work with
  - Correctness extremely hard to prove
  - Robustness hard to analyze
- Networking gurus have a vast knowledge of special cases that can lead to strange behaviors
  - Mis-configuration is a big cause of errors
- Soft state helps a lot, but wastes many messages!

Distributed Fixed Point Computation

We have n nodes and these nodes communicate by message passing. A node i takes a vector $x = (x_1, ..., x_n)$ (where the source of the i^{th} component is node i), $x \in X_i \in R$ and computes a value $f_i(x) \in X_i$. Then this value is sent to all of i's neighbors.

Letting $f(x) = (f_1(x), f_2(x), ..., f_n(x))$, we define x^* to be a fixed point if

$$x^* = f(x^*)$$

General Convergence Theorem

Suppose there exists a sequence of nested subsets of $X = X_1 \times X_2 \times X_3 \times \ldots \times X_n$, $\{X(k)\}$, $k = 0, 1, 2, \ldots$ such that

$$\ldots \subset X(k+1) \subset X(k) \subset \ldots \subset X$$

satisfying two intuitive conditions.

Then if the algorithm begins with an initial estimate $x(0) = (x_1(0), \ldots, x_n(0)) \in X(0)$, the totally asynchronous system will converge the fixed point of f, i.e. every limit point of $\{x(t)\}$ is a fixed point of f.

Conditions

1. Synchronous Convergence: For $x \in X(k)$ we have

$$f(x) \in X(k+1), \ \forall k.$$

Also, every limit point of a sequence $\{y^k\}$, where $y^k \in X(k)$ is also a fixed point of f.

Conditions

1. Synchronous Convergence: For $x \in X(k)$ we have

$$f(x) \in X(k+1), \ \forall k.$$

Also, every limit point of a sequence $\{y^k\}$, where $y^k \in X(k)$ is also a fixed point of f.

2. Box Condition: For every k there exists sets $X_i(k) \subset X$ such that:

$$X(k) = X_1(k) \times X_2(k) \times \ldots \times X_n(k).$$

This says that if x and $\bar{x} \in X(k)$ and we replace the i^{th} component of $\bar{x}$ with the i^{th} component of x, then the new vector is still in $X(k)$.

Special Case: Monotone Mappings

f is monontone: $x \leq y \Rightarrow f(x) \leq f(y)$.

Let x^* be a unique fixed point for f in X and suppose there exist, $\bar{x}$ and $\underline{x}$ such that

$$\underline{x} \leq f(\underline{x}) \leq f(\bar{x}) \leq \bar{x}$$

and

$$\lim_{t \to \infty} f^k(\underline{x}) = \lim_{t \to \infty} f^k(\bar{x}) = x^*.$$

Monotone Mappings Converge Asynchronously

Choose $X(k) = \{x : f^k(\underline{x}) \leq x \leq f^k(\bar{x})\}$

1. Synchronous Convergence holds since

$$f^k(\underline{x}) \leq f^{k+1}(\underline{x}) \leq x^* \leq f^{k+1}(\bar{x}) \leq f_k(\bar{x}).$$

2. Box Condition Holds trivially

So if we can find $\underline{x}$ and $\bar{x}$ we are done.

Bellman Ford

Set $f_1(x) = 0$ and

$$f_i(x) = \min_{j \in N(i)} \{d_{ij} + x_j\}.$$

f is monotone. Now choose $\underline{x}$ and $\bar{x}$:

For $i \neq 1$, $\underline{x}_i = x_i^* - \Delta$, $\Delta > 0$ and $\bar{x}_i = \infty$.

Then we have already shown that both of these vectors as initial conditions result in convergence to x^*.

Other systems for which the result holds

- See
  - Parallel and Distributed Computation by Dimitri Bertsekas and John Tsitsiklis, Prentice Hall 1989

Verdict on Distance Vector BF

- Requires no synchronization, works with limited topology information
- Doesn't deal well with changing topologies since it does not include "reachability" information
- Use path vectors --- send the shortest path not just the distance estimate.
  - Expensive fix!

Oscillations Revisited

Conclusions

- It is extremely difficult to design and verify correctness of distribute algorithms
  - But there is some (not enough) theory to help…
- Even when we decouple costs from link flow, route computation is far from straightforward
- Link State Protocols, combined with hierarchical routing work probably work better than distance vector approaches, but the jury is still out