Dynamic Networks: Robots, Agents, and Topology Manipulation

David Ko

April 30, 2009

Table of Contents

What is a mobile agent?

1. Special case of a general "software agent"
   - A piece of software
   - Autonomous
   - Intellegent
   - Acts "on behalf of" agency or other entity
2. Mobile agents are mobile: The code can migrate itself and portions of its execution state to other physical locations.

What is a mobile agent?

1. Special case of a general "software agent"
   - A piece of software
   - Autonomous
   - Intellegent
   - Acts "on behalf of" agency or other entity
2. Mobile agents are mobile: The code can migrate itself and portions of its execution state to other physical locations.
3. Mobile agents operate out of agencies. Agencies are software platforms that provide services and an execution environment to mobile agents.

What is a mobile agent?

1. Special case of a general "software agent"
   - A piece of software
   - Autonomous
   - Intellegent
   - Acts "on behalf of" agency or other entity
2. Mobile agents are mobile: The code can migrate itself and portions of its execution state to other physical locations.
3. Mobile agents operate out of agencies. Agencies are software platforms that provide services and an execution environment to mobile agents.

Agency Overview

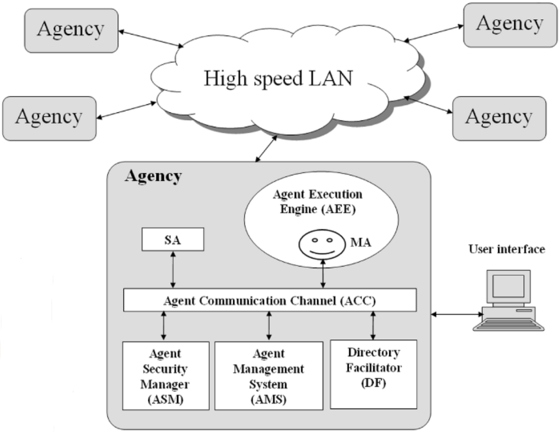


Figure: Agency Overview

Why use mobile agents?

Some commonly cited reasons to use the mobile-agent paradigm of computation: (from Wikipedia)

- Reduce network load: Processing on data may be done in place, rather than collecting data at a central location.
- Parallel processing: Multiple agents may cooperate operating on different hosts.

Why use mobile agents?

Some commonly cited reasons to use the mobile-agent paradigm of computation: (from Wikipedia)

- Reduce network load: Processing on data may be done in place, rather than collecting data at a central location.
- Parallel processing: Multiple agents may cooperate operating on different hosts.
- Dynamic adaptation: Agents may choose actions dependent on the state of their current host.

Why use mobile agents?

Some commonly cited reasons to use the mobile-agent paradigm of computation: (from Wikipedia)

- Reduce network load: Processing on data may be done in place, rather than collecting data at a central location.
- Parallel processing: Multiple agents may cooperate operating on different hosts.
- Dynamic adaptation: Agents may choose actions dependent on the state of their current host.
- Tolerant to network faults: Able to operate in a network with fluctuating connectivity.

Why use mobile agents?

Some commonly cited reasons to use the mobile-agent paradigm of computation: (from Wikipedia)

- Reduce network load: Processing on data may be done in place, rather than collecting data at a central location.
- Parallel processing: Multiple agents may cooperate operating on different hosts.
- Dynamic adaptation: Agents may choose actions dependent on the state of their current host.
- Tolerant to network faults: Able to operate in a network with fluctuating connectivity.
- Flexible Maintenance: To change an agent's actions, only one copy of source must be updated as opposed to each computational host which may host the agency.

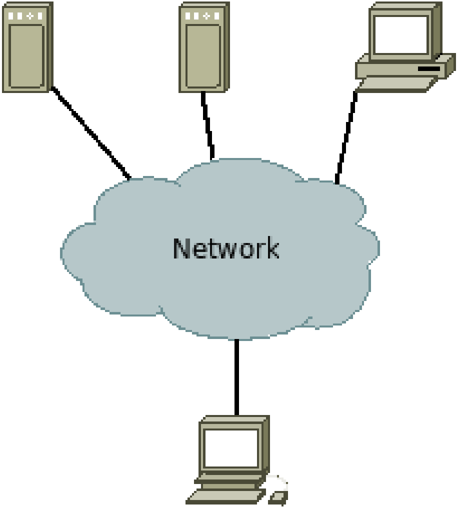
Why use mobile agents?

Some commonly cited reasons to use the mobile-agent paradigm of computation: (from Wikipedia)

- Reduce network load: Processing on data may be done in place, rather than collecting data at a central location.
- Parallel processing: Multiple agents may cooperate operating on different hosts.
- Dynamic adaptation: Agents may choose actions dependent on the state of their current host.
- Tolerant to network faults: Able to operate in a network with fluctuating connectivity.
- Flexible Maintenance: To change an agent's actions, only one copy of source must be updated as opposed to each computational host which may host the agency.

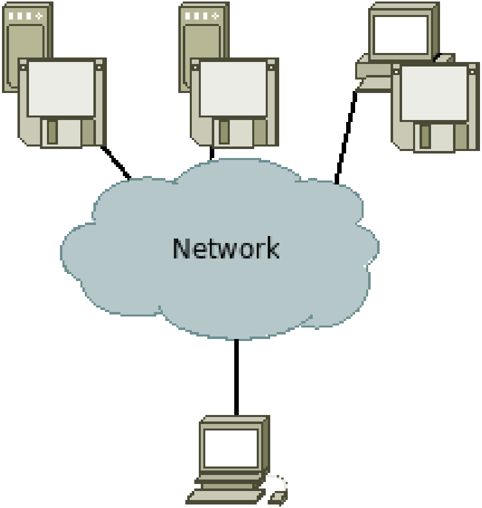
Data processing example

Traditional data allocation method:



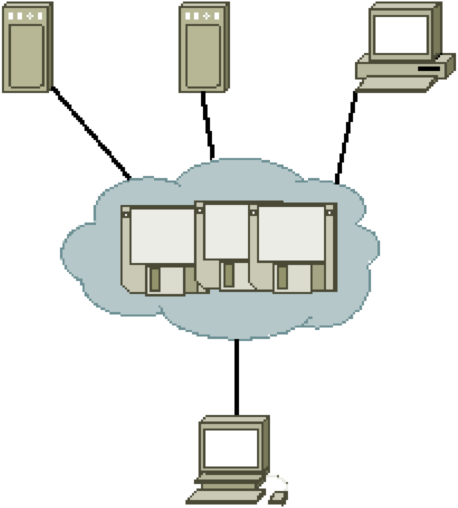
Data processing example

Traditional data allocation method:



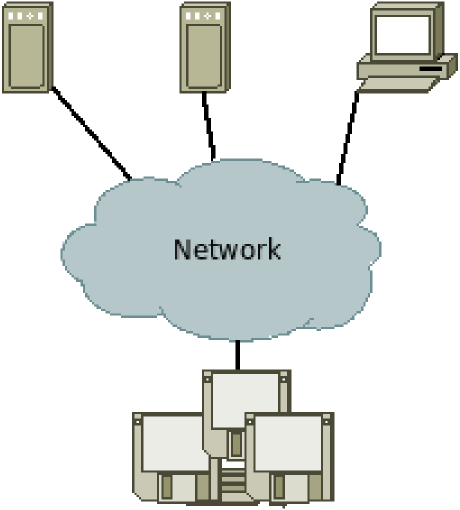
Data processing example

Traditional data allocation method:



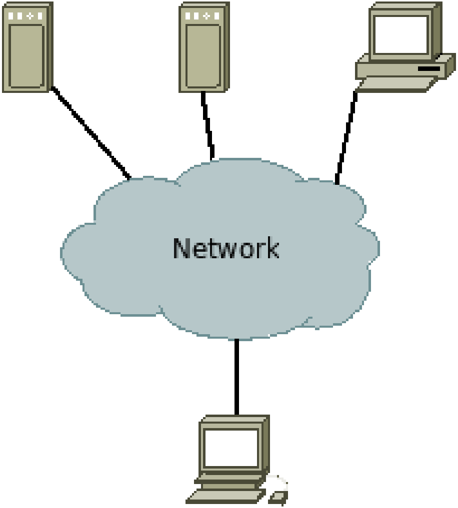
Data processing example

Traditional data allocation method:



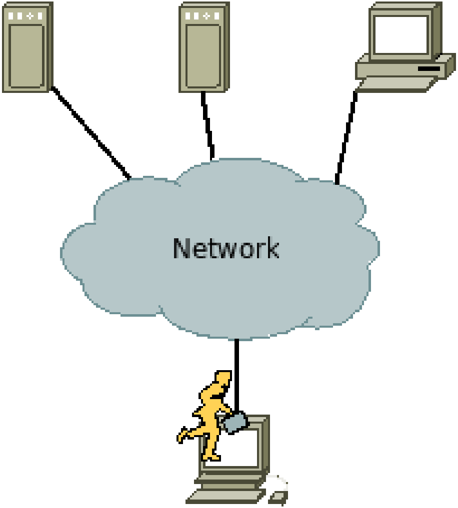
Data processing example

Agent based method:



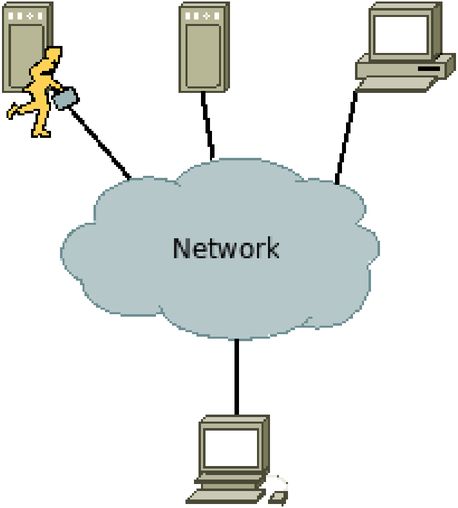
Data processing example

Agent based method:



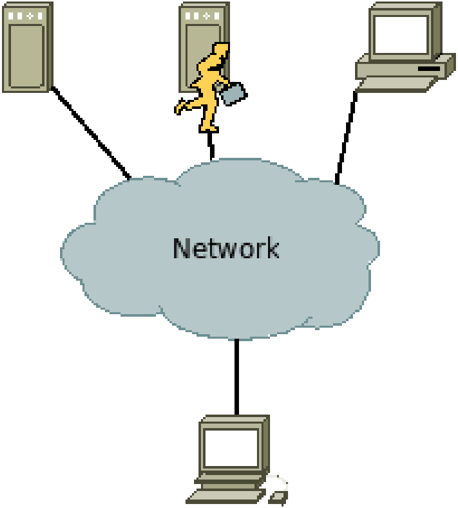
Data processing example

Agent based method:



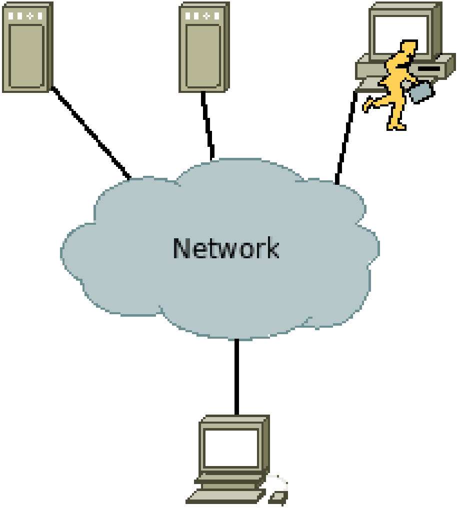
Data processing example

Agent based method:



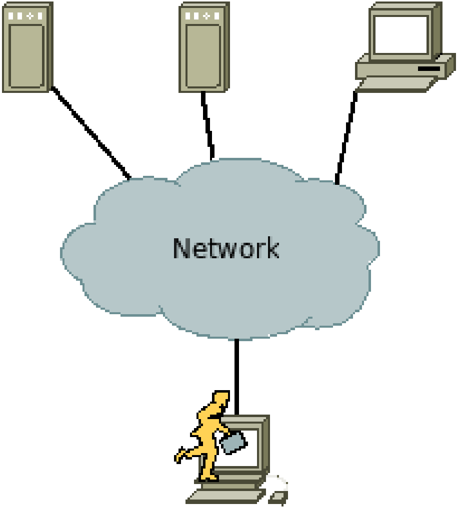
Data processing example

Agent based method:



Data processing example

Agent based method:



Why use mobile agents?

Mobile agents are being used and/or researched for use in the following fields:

- Telecom industry
- Manufacturing

Why use mobile agents?

Mobile agents are being used and/or researched for use in the following fields:

- Telecom industry
- Manufacturing
- Mapping

Why use mobile agents?

Mobile agents are being used and/or researched for use in the following fields:

- Telecom industry
- Manufacturing
- Mapping
- Data Mining

Why use mobile agents?

Mobile agents are being used and/or researched for use in the following fields:

- Telecom industry
- Manufacturing
- Mapping
- Data Mining
- Autonomous mobile robotics

Why use mobile agents?

Mobile agents are being used and/or researched for use in the following fields:

- Telecom industry
- Manufacturing
- Mapping
- Data Mining
- Autonomous mobile robotics

Java Agent Development Framework (JADE)

- Ported to mobile phones, palmtops, ... anything that supports Java.
- Agent programming paradigm is based on “behaviours”.

Java Agent Development Framework (JADE)

- Ported to mobile phones, palmtops, ... anything that supports Java.
- Agent programming paradigm is based on "behaviours".
  - Each agent typically has at least one behaviour.

Java Agent Development Framework (JADE)

- Ported to mobile phones, palmtops, ... anything that supports Java.
- Agent programming paradigm is based on “behaviours”.
  - Each agent typically has at least one behaviour.
  - Each agent may supports complex extendable behaviours which may run in parallel.

Java Agent Development Framework (JADE)

- Ported to mobile phones, palmtops, ... anything that supports Java.
- Agent programming paradigm is based on "behaviours".
  - Each agent typically has at least one behaviour.
  - Each agent may supports complex extendable behaviours which may run in parallel.
  - Behaviours may be triggered by events (receipt of a message, for example), they may be cyclic, or customized by the programmer.

Java Agent Development Framework (JADE)

- Ported to mobile phones, palmtops, ... anything that supports Java.
- Agent programming paradigm is based on "behaviours".
  - Each agent typically has at least one behaviour.
  - Each agent may supports complex extendable behaviours which may run in parallel.
  - Behaviours may be triggered by events (receipt of a message, for example), they may be cyclic, or customized by the programmer.
- Uses a centralized agency communication scheme. A single agency is designated the main container. All other containers communicate with the main one.

Java Agent Development Framework (JADE)

- Ported to mobile phones, palmtops, ... anything that supports Java.
- Agent programming paradigm is based on "behaviours".
  - Each agent typically has at least one behaviour.
  - Each agent may supports complex extendable behaviours which may run in parallel.
  - Behaviours may be triggered by events (receipt of a message, for example), they may be cyclic, or customized by the programmer.
- Uses a centralized agency communication scheme. A single agency is designated the main container. All other containers communicate with the main one.
- Designed for application level agent interaction and communication.

Java Agent Development Framework (JADE)

- Ported to mobile phones, palmtops, ... anything that supports Java.
- Agent programming paradigm is based on "behaviours".
  - Each agent typically has at least one behaviour.
  - Each agent may supports complex extendable behaviours which may run in parallel.
  - Behaviours may be triggered by events (receipt of a message, for example), they may be cyclic, or customized by the programmer.
- Uses a centralized agency communication scheme. A single agency is designated the main container. All other containers communicate with the main one.
- Designed for application level agent interaction and communication.
- Compliant with the Foundation for Intellegent Physical Agents (FIPA). (More on this later)

Java Agent Development Framework (JADE)

- Ported to mobile phones, palmtops, ... anything that supports Java.
- Agent programming paradigm is based on "behaviours".
  - Each agent typically has at least one behaviour.
  - Each agent may supports complex extendable behaviours which may run in parallel.
  - Behaviours may be triggered by events (receipt of a message, for example), they may be cyclic, or customized by the programmer.
- Uses a centralized agency communication scheme. A single agency is designated the main container. All other containers communicate with the main one.
- Designed for application level agent interaction and communication.
- Compliant with the Foundation for Intellegent Physical Agents (FIPA). (More on this later)

Mobile-C

- Mobile-C is an agency platform developed at the Integration Engineering Laboratory at UC Davis.
- Mobile-C supports C/C++ agents.

Mobile-C

- Mobile-C is an agency platform developed at the Integration Engineering Laboratory at UC Davis.
- Mobile-C supports C/C++ agents.
- Mobile-C is designed to be

Mobile-C

- Mobile-C is an agency platform developed at the Integration Engineering Laboratory at UC Davis.
- Mobile-C supports C/C++ agents.
- Mobile-C is designed to be
  - Lightweight

Mobile-C

- Mobile-C is an agency platform developed at the Integration Engineering Laboratory at UC Davis.
- Mobile-C supports C/C++ agents.
- Mobile-C is designed to be
  - Lightweight
  - Low level

Mobile-C

- Mobile-C is an agency platform developed at the Integration Engineering Laboratory at UC Davis.
- Mobile-C supports C/C++ agents.
- Mobile-C is designed to be
  - Lightweight
  - Low level
  - Interpretive

Mobile-C

- Mobile-C is an agency platform developed at the Integration Engineering Laboratory at UC Davis.
- Mobile-C supports C/C++ agents.
- Mobile-C is designed to be
  - Lightweight
  - Low level
  - Interpretive
- Ported to major operating systems and architectures, including the ARM architecture for tiny computers.

Mobile-C

- Mobile-C is an agency platform developed at the Integration Engineering Laboratory at UC Davis.
- Mobile-C supports C/C++ agents.
- Mobile-C is designed to be
  - Lightweight
  - Low level
  - Interpretive
- Ported to major operating systems and architectures, including the ARM architecture for tiny computers.
- Agent programming paradigm is functional, in the C language.

Mobile-C

- Mobile-C is an agency platform developed at the Integration Engineering Laboratory at UC Davis.
- Mobile-C supports C/C++ agents.
- Mobile-C is designed to be
  - Lightweight
  - Low level
  - Interpretive
- Ported to major operating systems and architectures, including the ARM architecture for tiny computers.
- Agent programming paradigm is functional, in the C language.
- Like JADE, Mobile-C is also compliant with FIPA standards.

Mobile-C

- Mobile-C is an agency platform developed at the Integration Engineering Laboratory at UC Davis.
- Mobile-C supports C/C++ agents.
- Mobile-C is designed to be
  - Lightweight
  - Low level
  - Interpretive
- Ported to major operating systems and architectures, including the ARM architecture for tiny computers.
- Agent programming paradigm is functional, in the C language.
- Like JADE, Mobile-C is also compliant with FIPA standards.

FIPA

1. FIPA is an organization founded in the mid 90's to help formulate standards for agent communication.
2. FIPA has written standards for communication message format as well as certain protocols.

FIPA

1. FIPA is an organization founded in the mid 90's to help formulate standards for agent communication.
2. FIPA has written standards for communication message format as well as certain protocols.
3. Since both Mobile-C and JADE are FIPA compliant, they may exchange messages with each other.

FIPA

1. FIPA is an organization founded in the mid 90's to help formulate standards for agent communication.
2. FIPA has written standards for communication message format as well as certain protocols.
3. Since both Mobile-C and JADE are FIPA compliant, they may exchange messages with each other.

FIPA

1. FIPA is an organization founded in the mid 90's to help formulate standards for agent communication.
2. FIPA has written standards for communication message format as well as certain protocols.
3. Since both Mobile-C and JADE are FIPA compliant, they may exchange messages with each other.

Some of my interests

- Mobile Agents.
- Autonomous mobile robots.

Some of my interests

- Mobile Agents.
- Autonomous mobile robots.
- Robot swarm behaviour.

Some of my interests

- Mobile Agents.
- Autonomous mobile robots.
- Robot swarm behaviour.

Some important aspects for an autonomous robot system to work:

- Autonomy.

Some of my interests

- Mobile Agents.
- Autonomous mobile robots.
- Robot swarm behaviour.

Some important aspects for an autonomous robot system to work:

- Autonomy.
- Robot Intellegence.

Some of my interests

- Mobile Agents.
- Autonomous mobile robots.
- Robot swarm behaviour.

Some important aspects for an autonomous robot system to work:

- Autonomy.
- Robot Intellegence.
- Swarm Intellegence.

Some of my interests

- Mobile Agents.
- Autonomous mobile robots.
- Robot swarm behaviour.

Some important aspects for an autonomous robot system to work:

- Autonomy.
- Robot Intellegence.
- Swarm Intellegence.
- Robot Cooperation.

Some of my interests

- Mobile Agents.
- Autonomous mobile robots.
- Robot swarm behaviour.

Some important aspects for an autonomous robot system to work:

- Autonomy.
- Robot Intellegence.
- Swarm Intellegence.
- Robot Cooperation.
- Fault Tolerance.

Some of my interests

- Mobile Agents.
- Autonomous mobile robots.
- Robot swarm behaviour.

Some important aspects for an autonomous robot system to work:

- Autonomy.
- Robot Intellegence.
- Swarm Intellegence.
- Robot Cooperation.
- Fault Tolerance.
  - Robustness of individual robots.

Some of my interests

- Mobile Agents.
- Autonomous mobile robots.
- Robot swarm behaviour.

Some important aspects for an autonomous robot system to work:

- Autonomy.
- Robot Intellegence.
- Swarm Intellegence.
- Robot Cooperation.
- Fault Tolerance.
  - Robustness of individual robots.
  - Robustness of swarm. Must have reliable communication!

Some of my interests

- Mobile Agents.
- Autonomous mobile robots.
- Robot swarm behaviour.

Some important aspects for an autonomous robot system to work:

- Autonomy.
- Robot Intellegence.
- Swarm Intellegence.
- Robot Cooperation.
- Fault Tolerance.
  - Robustness of individual robots.
  - Robustness of swarm. Must have reliable communication!

What applications are well suited for autonomous mobile robots?

Autonomous swarms of mobile robots are well suited to many applications including:

- Exploration applications.
  - Scientific, extra planetary, etc.

What applications are well suited for autonomous mobile robots?

Autonomous swarms of mobile robots are well suited to many applications including:

- Exploration applications.
    - Scientific, extra planetary, etc.
    - Search and rescue

What applications are well suited for autonomous mobile robots?

Autonomous swarms of mobile robots are well suited to many applications including:

- Exploration applications.
  - Scientific, extra planetary, etc.
  - Search and rescue
  - Military

What applications are well suited for autonomous mobile robots?

Autonomous swarms of mobile robots are well suited to many applications including:

- Exploration applications.
  - Scientific, extra planetary, etc.
  - Search and rescue
  - Military
- Data collection

What applications are well suited for autonomous mobile robots?

Autonomous swarms of mobile robots are well suited to many applications including:

- Exploration applications.
  - Scientific, extra planetary, etc.
  - Search and rescue
  - Military
- Data collection

In each case, robust robot communication is essential. The robot communication network must be fault tolerant.

What applications are well suited for autonomous mobile robots?

Autonomous swarms of mobile robots are well suited to many applications including:

- Exploration applications.
  - Scientific, extra planetary, etc.
  - Search and rescue
  - Military
- Data collection

In each case, robust robot communication is essential. The robot communication network must be fault tolerant.

Preliminary work done by Basu

- Proposed that in order to be fault tolerant, a robot network should be biconnected. In other words, the network should contain no cut vertices.
- Robot connectivity determined by limited communication range.
- Robots in a robot network may physically move to ensure a biconnected network.
- Cases presented by Basu include:
  - Linear contraction case.
  - Two dimensional contraction case.
  - Two dimensional block movement case.

Linear Contraction Case

Minimize

$$D_{total} = \sum_{i=1}^{N} |x_i - p_i| \tag{1}$$

With the following constraints:

$$x_1 \geq p_1 \tag{2}$$

$$x_N \leq p_N \tag{3}$$

$$x_i - x_{i-1} \geq 0, \ 2 \leq i \leq N \tag{4}$$

$$x_i - x_{i-2} \leq 1, \ 3 \leq i \leq N \tag{5}$$

2-D Contraction Case

2-D analog of the linear case.

1. Calculate “center of mass” of all robots.
2. Each robot begins to move towards the center at a speed proportional to its distance from the center. Those that are farther away move faster and vise versa.

2-D Contraction Case

2-D analog of the linear case.

1. Calculate “center of mass” of all robots.
2. Each robot begins to move towards the center at a speed proportional to its distance from the center. Those that are farther away move faster and vise versa.
3. Eventually, the network will become biconnected.

2-D Contraction Case

2-D analog of the linear case.

1. Calculate "center of mass" of all robots.
2. Each robot begins to move towards the center at a speed proportional to its distance from the center. Those that are farther away move faster and vise versa.
3. Eventually, the network will become biconnected.

2-D Block Movement Case

1. Divide graph into a bipartite graph of blocks and cut vertices.
2. Block with most nodes is deemed the root block.

2-D Block Movement Case

1. Divide graph into a bipartite graph of blocks and cut vertices.
2. Block with most nodes is deemed the root block.
3. Other blocks move toward the root block, thereby eliminating cut vertices.

2-D Block Movement Case

1. Divide graph into a bipartite graph of blocks and cut vertices.
2. Block with most nodes is deemed the root block.
3. Other blocks move toward the root block, thereby eliminating cut vertices.

2-D Block Movement Case

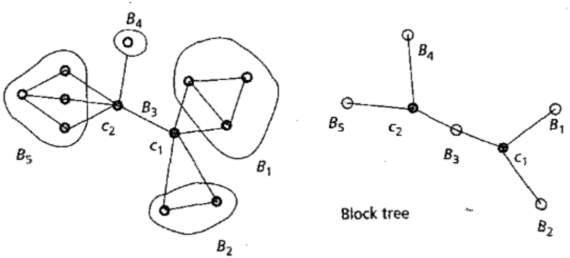


Figure: Basu's decomposition of a graph into a bipartite block tree.

Issues with Basu's algorithms

- Requires global knowledge of network topology.
- Assumes homogenous robots in a homogenous environment.

Issues with Basu's algorithms

- Requires global knowledge of network topology.
- Assumes homogenous robots in a homogenous environment.
  - 2-D contraction and block movement may leave edgebots behind due to speed fluctuations, limitations.

Issues with Basu's algorithms

- Requires global knowledge of network topology.
- Assumes homogenous robots in a homogenous environment.
  - 2-D contraction and block movement may leave edgebots behind due to speed fluctuations, limitations.
- 2-D block movement still moves many bots unnecessarily.

Issues with Basu's algorithms

- Requires global knowledge of network topology.
- Assumes homogenous robots in a homogenous environment.
  - 2-D contraction and block movement may leave edgebots behind due to speed fluctuations, limitations.
- 2-D block movement still moves many bots unnecessarily.
- 2-D block movement requires flocking. Flocking is not trivial!

Issues with Basu's algorithms

- Requires global knowledge of network topology.
- Assumes homogenous robots in a homogenous environment.
  - 2-D contraction and block movement may leave edgebots behind due to speed fluctuations, limitations.
- 2-D block movement still moves many bots unnecessarily.
- 2-D block movement requires flocking. Flocking is not trivial!
- Movement iteratively based on largest block.

Issues with Basu's algorithms

- Requires global knowledge of network topology.
- Assumes homogenous robots in a homogenous environment.
  - 2-D contraction and block movement may leave edgebots behind due to speed fluctuations, limitations.
- 2-D block movement still moves many bots unnecessarily.
- 2-D block movement requires flocking. Flocking is not trivial!
- Movement iteratively based on largest block.
  - Not only does the algorithm need to know the global topology at the beginning of the algorithm, it needs to recalculate it after every block movement.

Issues with Basu's algorithms

- Requires global knowledge of network topology.
- Assumes homogenous robots in a homogenous environment.
  - 2-D contraction and block movement may leave edgebots behind due to speed fluctuations, limitations.
- 2-D block movement still moves many bots unnecessarily.
- 2-D block movement requires flocking. Flocking is not trivial!
- Movement iteratively based on largest block.
  - Not only does the algorithm need to know the global topology at the beginning of the algorithm, it needs to recalculate it after every block movement.

A Better Algorithm

- Das et. al. propose a localized robot movement control algorithm to create fault tolerant robot networks.
- The algorithm is localized in the sense that each robot only knows/cares about its p-hop neighbors.

A Better Algorithm

- Das et. al. propose a localized robot movement control algorithm to create fault tolerant robot networks.
- The algorithm is localized in the sense that each robot only knows/cares about its p-hop neighbors.

p-hop neighbors

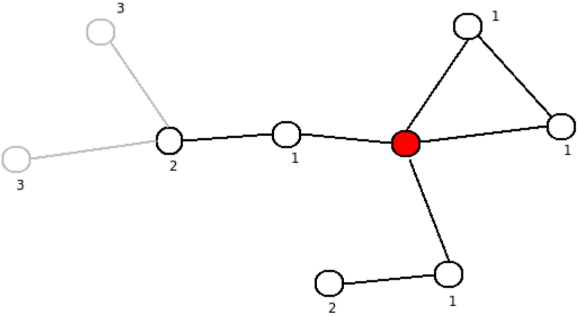


Figure: The 2-hop network of a certain node.

Critical Nodes

- If a node is a cut vertex within its p-hop subgraph, it is considered a critical node.
- All global cut vertices will be a p-hop critical node.

Critical Nodes

- If a node is a cut vertex within its p-hop subgraph, it is considered a critical node.
- All global cut vertices will be a p-hop critical node.
- Authors state that 80% of locally critical nodes are also globally critical. (For p=3)

Critical Nodes

- If a node is a cut vertex within its p-hop subgraph, it is considered a critical node.
- All global cut vertices will be a p-hop critical node.
- Authors state that 80% of locally critical nodes are also globally critical. (For p=3)

Critical Node Availability

- Authors define a critical node as being "available" if it has non-critical neighbors.
- Otherwise, the node is "non-available".

Critical Node Availability

- Authors define a critical node as being "available" if it has non-critical neighbors.
- Otherwise, the node is "non-available".
- A node is defined as a "critical head" if
  - It is available AND
  - it has the highest ID of its available critical neighbors OR it has no available critical neighbors

Critical Node Availability

- Authors define a critical node as being "available" if it has non-critical neighbors.
- Otherwise, the node is "non-available".
- A node is defined as a "critical head" if
  - It is available AND
  - it has the highest ID of its available critical neighbors OR it has no available critical neighbors

Network Fault Cases

The authors divide up the possible ways in which cut vertices may exist in a network into three seperate cases.

1. A critical node exists with no critical neighbors.
2. A critical node exists with one critical neighbor.

Network Fault Cases

The authors divide up the possible ways in which cut vertices may exist in a network into three seperate cases.

1. A critical node exists with no critical neighbors.
2. A critical node exists with one critical neighbor.
3. A critical node exists with many critical neighbors.

Network Fault Cases

The authors divide up the possible ways in which cut vertices may exist in a network into three seperate cases.

1. A critical node exists with no critical neighbors.
2. A critical node exists with one critical neighbor.
3. A critical node exists with many critical neighbors.

Case 1: A Critical Node with No Critical Neighbors

- Critical node chooses one node from each subgraph. (What if there are more than two subgraphs?)

Case 1: A Critical Node with No Critical Neighbors

- Critical node chooses one node from each subgraph. (What if there are more than two subgraphs?)
- Critical node commands the two selected nodes to move toward each other until they become neighbors.

Case 1: A Critical Node with No Critical Neighbors

- Critical node chooses one node from each subgraph. (What if there are more than two subgraphs?)
- Critical node commands the two selected nodes to move toward each other until they become neighbors.
- The original cut vertex is no longer a cut vertex. Topology information is shared to check if new critical nodes were formed.

Case 1: A Critical Node with No Critical Neighbors

- Critical node chooses one node from each subgraph. (What if there are more than two subgraphs?)
- Critical node commands the two selected nodes to move toward each other until they become neighbors.
- The original cut vertex is no longer a cut vertex. Topology information is shared to check if new critical nodes were formed.
- Iterate if there are still critical nodes remaining or if new critical nodes were created.

Case 1: A Critical Node with No Critical Neighbors

- Critical node chooses one node from each subgraph. (What if there are more than two subgraphs?)
- Critical node commands the two selected nodes to move toward each other until they become neighbors.
- The original cut vertex is no longer a cut vertex. Topology information is shared to check if new critical nodes were formed.
- Iterate if there are still critical nodes remaining or if new critical nodes were created.

Case 1: A Critical Node with No Critical Neighbors

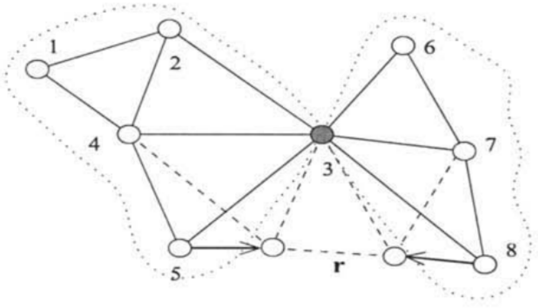


Figure: Node movement for a critical node with no critical neighbors.

Case 2: A Critical Node with One Critical Neighbor

- Critical nodes know about each other due to "critical announcement packets".
- Critical node with higher ID (aka Critical Head) takes charge. Critical node with lower ID idles.

Case 2: A Critical Node with One Critical Neighbor

- Critical nodes know about each other due to "critical announcement packets".
- Critical node with higher ID (aka Critical Head) takes charge. Critical node with lower ID idles.
- Critical Head divides local network into two seperate subgraphs, A and B, such that A contains the other critical node.

Case 2: A Critical Node with One Critical Neighbor

- Critical nodes know about each other due to "critical announcement packets".
- Critical node with higher ID (aka Critical Head) takes charge. Critical node with lower ID idles.
- Critical Head divides local network into two seperate subgraphs, A and B, such that A contains the other critical node.
- Critical Head directs a non-critical node in subgraph B toward the other critical node until they become neighbors.

Case 2: A Critical Node with One Critical Neighbor

- Critical nodes know about each other due to "critical announcement packets".
- Critical node with higher ID (aka Critical Head) takes charge. Critical node with lower ID idles.
- Critical Head divides local network into two seperate subgraphs, A and B, such that A contains the other critical node.
- Critical Head directs a non-critical node in subgraph B toward the other critical node until they become neighbors.

Case 2: A Critical Node with One Critical Neighbor

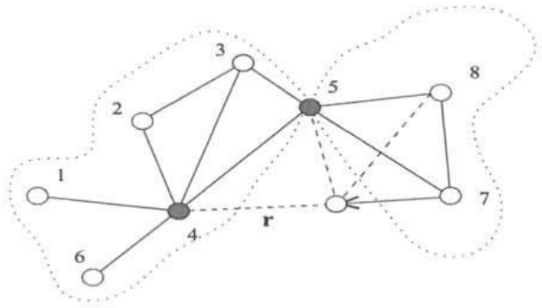


Figure: Node movement for a critical node with one critical neighbor.

Case 3: A Critical Node with Many Critical Neighbors

- Critical heads identify themselves.
- Critical heads choose a neighbor who is a critical node to pair with.

Case 3: A Critical Node with Many Critical Neighbors

- Critical heads identify themselves.
- Critical heads choose a neighbor who is a critical node to pair with.
- Critical heads invoke the Case 2 algorithm: Move a non-critical neighbor in the non-critical subgraph towards the chosen neighbor.

Case 3: A Critical Node with Many Critical Neighbors

- Critical heads identify themselves.
- Critical heads choose a neighbor who is a critical node to pair with.
- Critical heads invoke the Case 2 algorithm: Move a non-critical neighbor in the non-critical subgraph towards the chosen neighbor.

Case 3: A Critical Node with Many Critical Neighbors

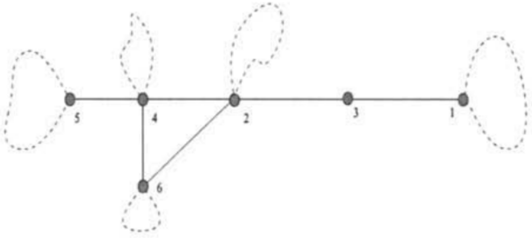


Figure: Node movement for a critical node with one critical neighbor.

Case 3: A Critical Node with Many Critical Neighbors

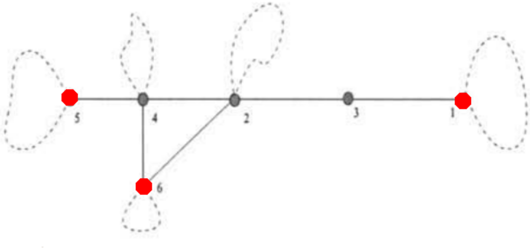


Figure: Node movement for a critical node with one critical neighbor.

Case 3: A Critical Node with Many Critical Neighbors

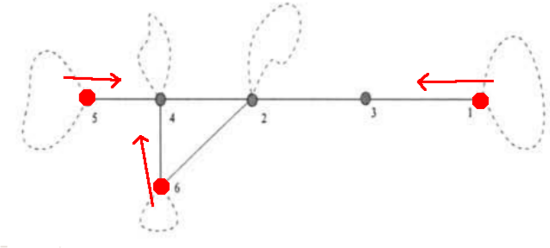


Figure: Node movement for a critical node with one critical neighbor.

Will there always be a critical head in a non-biconnected network?

- Lemma 1: There exists non-critical nodes in any connected network.
- Theorem 1: If the network is connected but not biconnected then it has a *either* a critical node with no critical neighbors *or* a critical head.

Will there always be a critical head in a non-biconnected network?

- Lemma 1: There exists non-critical nodes in any connected network.
- Theorem 1: If the network is connected but not biconnected then it has a *either* a critical node with no critical neighbors *or* a critical head.
- Proof:

Will there always be a critical head in a non-biconnected network?

- Lemma 1: There exists non-critical nodes in any connected network.
- Theorem 1: If the network is connected but not biconnected then it has a *either* a critical node with no critical neighbors *or* a critical head.
- Proof:
  - According to Lemma 1, there exist non-critical and critical nodes in the network.

Will there always be a critical head in a non-biconnected network?

- Lemma 1: There exists non-critical nodes in any connected network.
- Theorem 1: If the network is connected but not biconnected then it has a *either* a critical node with no critical neighbors *or* a critical head.
- Proof:
  - According to Lemma 1, there exist non-critical and critical nodes in the network.
  - Since the network is connected, some critical nodes must be connected to some non-critical nodes. These critical nodes are deemed “available”.

Will there always be a critical head in a non-biconnected network?

- Lemma 1: There exists non-critical nodes in any connected network.
- Theorem 1: If the network is connected but not biconnected then it has a *either* a critical node with no critical neighbors *or* a critical head.
- Proof:
  - According to Lemma 1, there exist non-critical and critical nodes in the network.
  - Since the network is connected, some critical nodes must be connected to some non-critical nodes. These critical nodes are deemed "available".
  - The available critical node with the largest ID must be by definition a critical head.

Will there always be a critical head in a non-biconnected network?

- Lemma 1: There exists non-critical nodes in any connected network.
- Theorem 1: If the network is connected but not biconnected then it has a *either* a critical node with no critical neighbors *or* a critical head.
- Proof:
  - According to Lemma 1, there exist non-critical and critical nodes in the network.
  - Since the network is connected, some critical nodes must be connected to some non-critical nodes. These critical nodes are deemed "available".
  - The available critical node with the largest ID must be by definition a critical head.

Why is this algorithm cool?

- The algorithm is locally executed. It is a peer to peer algorithm, similar to early versions of Gnutella. There is no need for global topology knowledge.
- Fixing false positive cut vertices may not be a bad thing. If a node is locally critical, it may be worthwile to reinforce it.

Why is this algorithm cool?

- The algorithm is locally executed. It is a peer to peer algorithm, similar to early versions of Gnutella. There is no need for global topology knowledge.
- Fixing false positive cut vertices may not be a bad thing. If a node is locally critical, it may be worthwile to reinforce it.

Issues? Future work ideas?

- Message flooding
- Message oscillation

Issues? Future work ideas?

- Message flooding
- Message oscillation
- Each bot needs to know physical location of all its p-hop neighbors. GPS? Triangulation?

Issues? Future work ideas?

- Message flooding
- Message oscillation
- Each bot needs to know physical location of all its p-hop neighbors. GPS? Triangulation?
- Error handling? What if a bot can't move as directed? Will a critical bot wait forever for a node to move?

Issues? Future work ideas?

- Message flooding
- Message oscillation
- Each bot needs to know physical location of all its p-hop neighbors. GPS? Triangulation?
- Error handling? What if a bot can't move as directed? Will a critical bot wait forever for a node to move?
- How representative is the random initial placement of bots in the simulation to real life? In the real world, bots may be deployed from a single point. Will they ever reach a "random" distribution?

Issues? Future work ideas?

- Message flooding
- Message oscillation
- Each bot needs to know physical location of all its p-hop neighbors. GPS? Triangulation?
- Error handling? What if a bot can't move as directed? Will a critical bot wait forever for a node to move?
- How representative is the random initial placement of bots in the simulation to real life? In the real world, bots may be deployed from a single point. Will they ever reach a "random" distribution?
- What if bots deployed from a single location approach a ring type topology? Then will every bot think they are a critical node? Lemma 1 and Theorem 1 consider the entire network, not the p-hop subgraph!

Issues? Future work ideas?

- Message flooding
- Message oscillation
- Each bot needs to know physical location of all its p-hop neighbors. GPS? Triangulation?
- Error handling? What if a bot can't move as directed? Will a critical bot wait forever for a node to move?
- How representative is the random initial placement of bots in the simulation to real life? In the real world, bots may be deployed from a single point. Will they ever reach a "random" distribution?
- What if bots deployed from a single location approach a ring type topology? Then will every bot think they are a critical node? Lemma 1 and Theorem 1 consider the entire network, not the p-hop subgraph!

New research ideas

- Agents are able to migrate from bot to bot
- What if we designate a small number of agents to be independent "leader" agents who can intellegently move other bots around at will with no regard for topology? What if we could migrate the leader agents to the edge of the network? Could we then have the leader bots direct the entire network by "dragging" it around? Essentially, what if critical nodes *are* permitted to move?

New research ideas

- Agents are able to migrate from bot to bot
- What if we designate a small number of agents to be independent "leader" agents who can intellegently move other bots around at will with no regard for topology? What if we could migrate the leader agents to the edge of the network? Could we then have the leader bots direct the entire network by "dragging" it around? Essentially, what if critical nodes *are* permitted to move?

Leader Agent/Bot

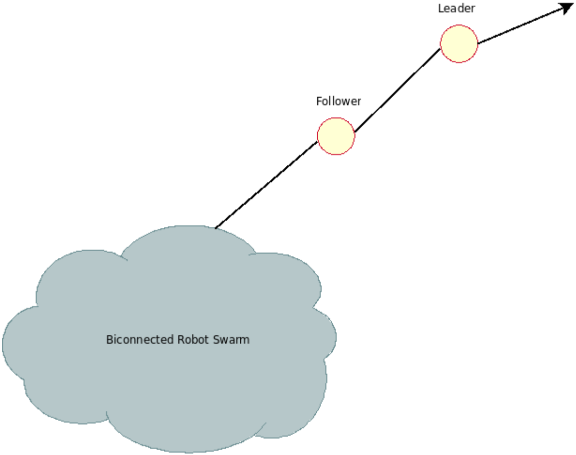


Figure: Conceptual drawing of a leader bot “dragging” a biconnected robot network.

The End

Questions? Comments?
Thanks for your time!