83-860-01
Algorithms in Ad-Hoc and Sensor Networks

Prof. Adrian Segall
Adrian.Segall@biu.ac.il

The seminar will consist of several introductory lectures on two topics:

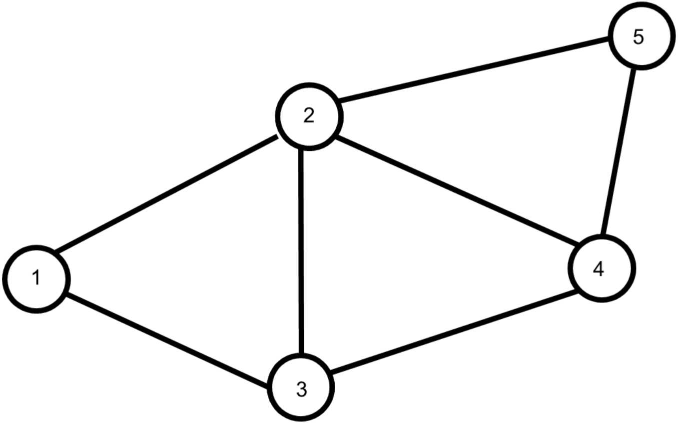
- Distributed Network Protocols (material: Slides and Lecture Notes)
- Ad-Hoc Protocols (material: Slides)

During these first weeks, students will select papers from a list, present them in the following weeks and submit a written summary of the presented paper(at most two pages).

Distributed Network Protocols

Prof. Adrian Segall
Department of Electrical Engineering
Technion, Israel Institute of Technology
segall at ee.technion.ac.il
and
Department of Computer Engineering
Bar Ilan University
Adrian.Segall at biu.ac.il

General "Classical" Network



Link Model

The Link may be in one of two *states* at each DLC:

- **Connected**
- **Initialization**

Only in *Connected* it can receive and send data frames. When entering *Initialization*, it resets counters and discards any unacknowledged data frames.

DLC protocol that ensures data reliability

Definition: A bit-oriented DLC procedure is said to ensure *data reliability* if it satisfies the following properties:

DLC protocol that ensures data reliability

Definition: A bit-oriented DLC procedure is said to ensure *data reliability* if it satisfies the following properties:

Follow-up: If a DLC enters Initialization Mode at some time when the other DLC is in Connected state, then the latter will also enter Initialization Mode in finite time.

DLC protocol that ensures data reliability

Definition: A bit-oriented DLC procedure is said to ensure *data reliability* if it satisfies the following properties:

- Follow-up: If a DLC enters Initialization Mode at some time when the other DLC is in Connected state, then the latter will also enter Initialization Mode in finite time.
- Crossing: If a DLC enters Initialization Mode at some time t_1, there is a time t after t_1 but before the DLC next enters Connected State, such that the other DLC is also in Initialization Mode and no packet accepted by the sender DLC at either end before time t can be delivered to the corresponding data sink after time t.

DLC protocol that ensures data reliability

Definition: A bit-oriented DLC procedure is said to ensure *data reliability* if it satisfies the following properties:

Follow-up: If a DLC enters Initialization Mode at some time when the other DLC is in Connected state, then the latter will also enter Initialization Mode in finite time.

Crossing: If a DLC enters Initialization Mode at some time t_1, there is a time t after t_1 but before the DLC next enters Connected State, such that the other DLC is also in Initialization Mode and no packet accepted by the sender DLC at either end before time t can be delivered to the corresponding data sink after time t.

Deadlock-Free: There exists a value T_1 such that if (a) both DLC's are in Initialization Mode at some time t and (b) during the interval of length T_1 after t there are no channel errors and (c) the delay for all frames (queueing+propagation) is bounded, then at time $t + T_1$ both DLC's are in Connected State. The DLC's stay in Connected State if there are no media failures afterwards.

DLC protocol that ensures data reliability

Definition: A bit-oriented DLC procedure is said to ensure *data reliability* if it satisfies the following properties:

Follow-up: If a DLC enters Initialization Mode at some time when the other DLC is in Connected state, then the latter will also enter Initialization Mode in finite time.

Crossing: If a DLC enters Initialization Mode at some time t_1, there is a time t after t_1 but before the DLC next enters Connected State, such that the other DLC is also in Initialization Mode and no packet accepted by the sender DLC at either end before time t can be delivered to the corresponding data sink after time t.

Deadlock-Free: There exists a value T_1 such that if (a) both DLC's are in Initialization Mode at some time t and (b) during the interval of length T_1 after t there are no channel errors and (c) the delay for all frames (queueing+propagation) is bounded, then at time $t + T_1$ both DLC's are in Connected State. The DLC's stay in Connected State if there are no media failures afterwards.

FIFO: Suppose that a DLC delivers to its data sink a packet that has been accepted at time t by the other DLC from the corresponding data source. Then all data packets accepted by the other DLC since it last entered Connected Mode until time t, have been delivered to the data sink without errors, in order, with no gaps or duplicates.

DLC protocol that ensures data reliability

Definition: A bit-oriented DLC procedure is said to ensure *data reliability* if it satisfies the following properties:

Follow-up: If a DLC enters Initialization Mode at some time when the other DLC is in Connected state, then the latter will also enter Initialization Mode in finite time.

Crossing: If a DLC enters Initialization Mode at some time t_1, there is a time t after t_1 but before the DLC next enters Connected State, such that the other DLC is also in Initialization Mode and no packet accepted by the sender DLC at either end before time t can be delivered to the corresponding data sink after time t.

Deadlock-Free: There exists a value T_1 such that if (a) both DLC's are in Initialization Mode at some time t and (b) during the interval of length T_1 after t there are no channel errors and (c) the delay for all frames (queueing+propagation) is bounded, then at time $t + T_1$ both DLC's are in Connected State. The DLC's stay in Connected State if there are no media failures afterwards.

FIFO: Suppose that a DLC delivers to its data sink a packet that has been accepted at time t by the other DLC from the corresponding data source. Then all data packets accepted by the other DLC since it last entered Connected Mode until time t, have been delivered to the data sink without errors, in order, with no gaps or duplicates.

Confirm: Whenever a DLC is in Connected State, all packets accepted from its data source since it last entered the Connected State, and considered acknowledged, have been delivered to the corresponding data sink.

DLC protocol that ensures data reliability

Definition: A bit-oriented DLC procedure is said to ensure *data reliability* if it satisfies the following properties:

- Follow-up: If a DLC enters Initialization Mode at some time when the other DLC is in Connected state, then the latter will also enter Initialization Mode in finite time.
- Crossing: If a DLC enters Initialization Mode at some time t_1, there is a time t after t_1 but before the DLC next enters Connected State, such that the other DLC is also in Initialization Mode and no packet accepted by the sender DLC at either end before time t can be delivered to the corresponding data sink after time t.
- Deadlock-Free: There exists a value T_1 such that if (a) both DLC's are in Initialization Mode at some time t and (b) during the interval of length T_1 after t there are no channel errors and (c) the delay for all frames (queueing+propagation) is bounded, then at time $t + T_1$ both DLC's are in Connected State. The DLC's stay in Connected State if there are no media failures afterwards.
- FIFO: Suppose that a DLC delivers to its data sink a packet that has been accepted at time t by the other DLC from the corresponding data source. Then all data packets accepted by the other DLC since it last entered Connected Mode until time t, have been delivered to the data sink without errors, in order, with no gaps or duplicates.
- Confirm: Whenever a DLC is in Connected State, all packets accepted from its data source since it last entered the Connected State, and considered acknowledged, have been delivered to the corresponding data sink.
- Delivery: Suppose that a DLC enters Connected State and stays there forever afterwards. Then all packets produced by that DLC's data source and accepted by the DLC after it entered Connected state are considered acknowledged within finite time.

Discussion:

Discussion:

Follow-up: the need for this property is obvious. In particular it disallows the situation where one DLC stays forever in Connected state and the other is in Initialization Mode.

Discussion:

Follow-up: the need for this property is obvious. In particular it disallows the situation where one DLC stays forever in Connected state and the other is in Initialization Mode.

Crossing relaxes and formalizes the usual notion of a *correct global initial state* where both DLC's are in Connected State with sequence number 0 and the channel is empty of frames. The generalization takes into consideration the case when one DLC enters Connected State and starts sending frames before the other enters Connected State, so that strictly speaking there is no instant when the system is in a *correct global initial state*. In this situation we still think of the DLC procedure as reliable, provided it satisfies the property indicated above under **Crossing**.

Discussion:

Follow-up: the need for this property is obvious. In particular it disallows the situation where one DLC stays forever in Connected state and the other is in Initialization Mode.

Crossing relaxes and formalizes the usual notion of a *correct global initial state* where both DLC's are in Connected State with sequence number 0 and the channel is empty of frames. The generalization takes into consideration the case when one DLC enters Connected State and starts sending frames before the other enters Connected State, so that strictly speaking there is no instant when the system is in a *correct global initial state*. In this situation we still think of the DLC procedure as reliable, provided it satisfies the property indicated above under **Crossing**.

Deadlock-Free says that if the channel works properly, the DLC's are not deadlocked in Initialization Mode.

Discussion:

Follow-up: the need for this property is obvious. In particular it disallows the situation where one DLC stays forever in Connected state and the other is in Initialization Mode.

Crossing relaxes and formalizes the usual notion of a *correct global initial state* where both DLC's are in Connected State with sequence number 0 and the channel is empty of frames. The generalization takes into consideration the case when one DLC enters Connected State and starts sending frames before the other enters Connected State, so that strictly speaking there is no instant when the system is in a *correct global initial state*. In this situation we still think of the DLC procedure as reliable, provided it satisfies the property indicated above under **Crossing**.

Deadlock-Free says that if the channel works properly, the DLC's are not deadlocked in Initialization Mode.

FIFO states that the sequence of packets delivered to the data sink is a prefix of the sequence received from the data source.

Discussion:

Follow-up: the need for this property is obvious. In particular it disallows the situation where one DLC stays forever in Connected state and the other is in Initialization Mode.

Crossing relaxes and formalizes the usual notion of a *correct global initial state* where both DLC's are in Connected State with sequence number 0 and the channel is empty of frames. The generalization takes into consideration the case when one DLC enters Connected State and starts sending frames before the other enters Connected State, so that strictly speaking there is no instant when the system is in a *correct global initial state*. In this situation we still think of the DLC procedure as reliable, provided it satisfies the property indicated above under **Crossing**.

Deadlock-Free says that if the channel works properly, the DLC's are not deadlocked in Initialization Mode.

FIFO states that the sequence of packets delivered to the data sink is a prefix of the sequence received from the data source.

Confirm states that packets that are considered acknowledged by the source DLC have indeed been delivered to the data sink.

Discussion:

Follow-up: the need for this property is obvious. In particular it disallows the situation where one DLC stays forever in Connected state and the other is in Initialization Mode.

Crossing relaxes and formalizes the usual notion of a *correct global initial state* where both DLC's are in Connected State with sequence number 0 and the channel is empty of frames. The generalization takes into consideration the case when one DLC enters Connected State and starts sending frames before the other enters Connected State, so that strictly speaking there is no instant when the system is in a *correct global initial state*. In this situation we still think of the DLC procedure as reliable, provided it satisfies the property indicated above under **Crossing**.

Deadlock-Free says that if the channel works properly, the DLC's are not deadlocked in Initialization Mode.

FIFO states that the sequence of packets delivered to the data sink is a prefix of the sequence received from the data source.

Confirm states that packets that are considered acknowledged by the source DLC have indeed been delivered to the data sink.

Delivery ensures that the DLC procedure is not the cause for non-delivery of data. It does not allow the possibility that the media is operational and is not declared failed by the failure detection mechanism, but the DLC procedure is stagnated in a situation where packets are not delivered or not considered acknowledged.

Discussion - cont'd

- **Delivery** and **Confirm** ensure that under the conditions stated in the Delivery property, all packets are delivered to the data sink in finite time.
- **FIFO** and **Confirm** ensure proper delivery of packets corresponding to frames that are *considered acknowledged*. At any instant there are $(W-1)$ packets that have been accepted from the data source but are not yet acknowledged. Such packets may or may not be delivered to the data sink (if the DLC enters Initialization Mode), but the **FIFO** property says that whatever is delivered to the sink, is delivered in sequence, whether it is considered acknowledged or not.

Network protocols: The Fixed Topology Model

a) Each link is bidirectional; the link connecting node i with node j considered in the direction from i to j is denoted (i, j).

Network protocols: The Fixed Topology Model

a) Each link is bidirectional; the link connecting node i with node j considered in the direction from i to j is denoted (i, j).

b) All messages are *control messages* of the DNP. We observe that those messages are considered as data packets by the DLC's on the links.

Network protocols: The Fixed Topology Model

a) Each link is bidirectional; the link connecting node i with node j considered in the direction from i to j is denoted (i, j).
b) All messages are *control messages* of the DNP. We observe that those messages are considered as data packets by the DLC's on the links.
c) Associated with each link, there is a Data-Link Control protocol that ensures Data Reliability. Since links and nodes do not fail in this model, Data Reliability means FIFO, Confirm and Delivery.

Network protocols: The Fixed Topology Model

a) Each link is bidirectional; the link connecting node i with node j considered in the direction from i to j is denoted (i, j).
b) All messages are *control messages* of the DNP. We observe that those messages are considered as data packets by the DLC's on the links.
c) Associated with each link, there is a Data-Link Control protocol that ensures Data Reliability. Since links and nodes do not fail in this model, Data Reliability means FIFO, Confirm and Delivery.
d) All messages received at a node i are stamped with the identification of the link from which they came and then transferred into a common queue; each node uses one processor for the purpose of the algorithm; the processor extracts the control message at the head of the queue (at that time we say that the node *receives* the message), proceeds to process it and discards the message when processing is completed; actions triggered by receipt of a message are atomic, namely no other operation related to the protocol is performed by the processor while a message is being processed; consequently we may relate all processing that takes place in response to the receipt of a control message to the instant this processing is completed and regard the processing as if it takes zero time.

e) Each node has an identification; before the protocol starts, each node knows the identity of all nodes that are potentially in the network; except when otherwise stated, it knows nothing about the topology of the network and in particular about what nodes actually belong to the network. We denote by $1, 2, ..., |\overline{V}|$ the nodes that are potentially in the network and by $1, 2, ..., |V|$ the nodes actually belonging to the network. We denote by $|E|$ the number of bidirectional links in the network and by $|\overline{E}|$ the number of links potentially in the network.

e) Each node has an identification; before the protocol starts, each node knows the identity of all nodes that are potentially in the network; except when otherwise stated, it knows nothing about the topology of the network and in particular about what nodes actually belong to the network. We denote by $1, 2, ..., |\overline{V}|$ the nodes that are potentially in the network and by $1, 2, ..., |V|$ the nodes actually belonging to the network. We denote by $|E|$ the number of bidirectional links in the network and by $|\overline{E}|$ the number of links potentially in the network.

f) Each node knows its adjacent links, and possibly the identity of its neighbors. The latter can be normally provided by the DLC Link Initialization at the time when the link is brought up. The collection of all neighbors of node i will be denoted by G_i.

e) Each node has an identification; before the protocol starts, each node knows the identity of all nodes that are potentially in the network; except when otherwise stated, it knows nothing about the topology of the network and in particular about what nodes actually belong to the network. We denote by $1, 2, ..., |\overline{V}|$ the nodes that are potentially in the network and by $1, 2, ..., |V|$ the nodes actually belonging to the network. We denote by $|E|$ the number of bidirectional links in the network and by $|\overline{E}|$ the number of links potentially in the network.

f) Each node knows its adjacent links, and possibly the identity of its neighbors. The latter can be normally provided by the DLC Link Initialization at the time when the link is brought up. The collection of all neighbors of node i will be denoted by G_i.

g) In some cases, the protocol may be started by only one node and in some others by several nodes asynchronously. This will be stated explicitly in the description of each protocol. A node starts the algorithm by receiving a special message *START* from the outside world; a standing assumption is that, once a node has entered the algorithm, it cannot receive *START*.

e) Each node has an identification; before the protocol starts, each node knows the identity of all nodes that are potentially in the network; except when otherwise stated, it knows nothing about the topology of the network and in particular about what nodes actually belong to the network. We denote by $1, 2, ..., |\overline{V}|$ the nodes that are potentially in the network and by $1, 2, ..., |V|$ the nodes actually belonging to the network. We denote by $|E|$ the number of bidirectional links in the network and by $|\overline{E}|$ the number of links potentially in the network.

f) Each node knows its adjacent links, and possibly the identity of its neighbors. The latter can be normally provided by the DLC Link Initialization at the time when the link is brought up. The collection of all neighbors of node i will be denoted by G_i.

g) In some cases, the protocol may be started by only one node and in some others by several nodes asynchronously. This will be stated explicitly in the description of each protocol. A node starts the algorithm by receiving a special message *START* from the outside world; a standing assumption is that, once a node has entered the algorithm, it cannot receive *START.*

h) *(don't postpone)* The message delay on a given link is measured from the time when the message is accepted by the DLC until it is delivered by the DLC at the other end to the Network Protocol. The message delays on a given link are assumed to be *strictly positive* and may be time varying, *with the restriction that always a message sent at a later time on a given link arrives at a later time*.

The Variable Topology Model

a'),c') Associated with each link there is a Data-Link protocol that ensures Data reliability, i.e. Follow-up, Deadlock-Free, Crossing, FIFO, Confirm and Delivery.

The Variable Topology Model

a'),c') Associated with each link there is a Data-Link protocol that ensures Data reliability, i.e. Follow-up, Deadlock-Free, Crossing, FIFO, Confirm and Delivery.

e') A link is considered to belong to the network, i.e. to be in the set E if both its ends are in Connected state for this link. Therefore $(i, l) \in E$ if and only if both $i \in G_l$ and $l \in G_i$.

The Variable Topology Model

a'),c') Associated with each link there is a Data-Link protocol that ensures Data reliability, i.e. Follow-up, Deadlock-Free, Crossing, FIFO, Confirm and Delivery.

e') A link is considered to belong to the network, i.e. to be in the set E if both its ends are in Connected state for this link. Therefore $(i, l) \in E$ if and only if both $i \in G_l$ and $l \in G_i$.

i) When a node comes up, it first performs the actions required by the Network Protocol and then proceeds to perform the Link Initialization Protocol for each of its links.

Basic Protocols - Propagation of Information

Protocol PI1

Messages

$MSG(info)$ - message carrying the information *info* to be propagated

Variables

G_i - set of neighbors of i

m_i - shows whether node i has already entered the protocol (values 0,1).

Initialization

if i receives a MSG, then

- just before receiving the first MSG, holds $m_i = 0$

Algorithm for node i

```
A1    receive MSG(info) from l ∈ G_i ∪ {nil}
A2    {   if  (m_i = 0) phase1();
      }
B1    phase1()
B2    {   m_i ← 1;
B3        accept(info);
B4        for  (k ∈ G_i) send MSG(info) to k;
      }
```

Properties of *PI1*

Theorem

Suppose that in Protocol $PI1$, node $s \in V$ receives $START$. Recall that $START$ is defined as the event when s receives MSG from nil. Then:

a) *All nodes $i \in V$ will accept the information in finite time and exactly once.*

Properties of *PI1*

Theorem

Suppose that in Protocol $PI1$, *node* $s \in V$ *receives* $START$. *Recall that* $START$ *is defined as the event when* s *receives* MSG *from* nil. *Then:*

a) *All nodes* $i \in V$ *will accept the information in finite time and exactly once.*
b) *During the execution of the protocol, exactly one* MSG *is sent on each link in each direction.*

Properties of *PI1*

Theorem

Suppose that in Protocol $PI1$, node $s \in V$ receives $START$. Recall that $START$ is defined as the event when s receives MSG from nil. Then:

a) *All nodes $i \in V$ will accept the information in finite time and exactly once.*
b) *During the execution of the protocol, exactly one MSG is sent on each link in each direction.*
c) *The propagation of information is the fastest possible, in the sense that no other protocol can bring the information to any node i faster than $PI1$.*

Properties of *PI1*

Theorem

Suppose that in Protocol $PI1$, node $s \in V$ receives $START$. Recall that $START$ is defined as the event when s receives MSG from nil. Then:

a) *All nodes $i \in V$ will accept the information in finite time and exactly once.*
b) *During the execution of the protocol, exactly one MSG is sent on each link in each direction.*
c) *The propagation of information is the fastest possible, in the sense that no other protocol can bring the information to any node i faster than $PI1$.*
d) *Define a* string of messages *as a sequence of messages (of some other protocol), such that each message except the first one is sent by a node i to some neighbor at or after the time when the previous message in the sequence was received by i from some neighbor. Then no string of messages can overtake $PI1$, i.e. if the originator of the string sends the first message in the string after it has entered $PI1$, then all messages in the string are received after the respective nodes have entered the $PI1$.*

Properties of *PI1*

Theorem

Suppose that in Protocol $PI1$, node $s \in V$ receives $START$. Recall that $START$ is defined as the event when s receives MSG from nil. Then:

a) *All nodes $i \in V$ will accept the information in finite time and exactly once.*
b) *During the execution of the protocol, exactly one MSG is sent on each link in each direction.*
c) *The propagation of information is the fastest possible, in the sense that no other protocol can bring the information to any node i faster than $PI1$.*
d) *Define a* string of messages *as a sequence of messages (of some other protocol), such that each message except the first one is sent by a node i to some neighbor at or after the time when the previous message in the sequence was received by i from some neighbor. Then no string of messages can overtake $PI1$, i.e. if the originator of the string sends the first message in the string after it has entered $PI1$, then all messages in the string are received after the respective nodes have entered the $PI1$.*

Note: Observe that properties c) and d) are similar, but not identical. Property c) says that no node can gain in terms of speed if $PI1$ is **replaced** by another protocol. Property d) says that if **both** $PI1$ and another protocol that generates *strings of messages* **operate** in the network, then no string of the other protocol can overtake $PI1$.

PI2

In PI2, we save some messages by having i not sending a message to p_i.

Protocol PI2

Messages

$MSG(info)$ - message carrying the information *info* to be propagated

Variables

G_i - set of neighbors of i

m_i - shows whether node i has already entered the protocol (values 0,1).

p_i - neighbor from which the first MSG is received

Initialization

if i receives a MSG, then

- just before receiving the first MSG, holds $m_i = 0$

Algorithm for node i

A1 receive $MSG(info)$ from $l \in G_i \cup \{nil\}$

A2 $\{$ *if* $(m_i = 0)$ $phase1()$;

$\}$

B1 $phase1()$

B2 $\{$ $m_i \leftarrow 1$;

B3 $p_i \leftarrow l$;

B4 accept($info$);

B5 *for* $(k \in G_i - \{p_i\})$ send *MSG(info)* to k;

$\}$

Properties of *PI2*

Theorem

Suppose that in Protocol $PI2$*, a node* $s \in V$ *receives* $START$*. Then:*

Properties of *PI2*

Theorem

Suppose that in Protocol PI2, a node $s \in V$ receives START. Then:

a) *all nodes $i \in V$ will accept the information in finite time and exactly once; after this happens, the links $\{(i, p_i)\ ,\ \forall i \in V\}$ will form a directed spanning tree rooted at s; in addition, for all i holds $t(phase1()_i) > t(phase1()_{p_i})$.*

Properties of *PI2*

Theorem

Suppose that in Protocol PI2, a node $s \in V$ receives START. Then:

a) *all nodes $i \in V$ will accept the information in finite time and exactly once; after this happens, the links $\{(i, p_i)\ ,\ \forall i \in V\}$ will form a directed spanning tree rooted at s; in addition, for all i holds $t(phase1()_i) > t(phase1()_{p_i})$.*

b) *During the execution of the protocol, exactly one MSG is sent on each link of the type $\neq (i, p_i)$, in each direction. On links of the type (i, p_i), a MSG is sent only in the direction from p_i to i.*

Properties of *PI2*

Theorem

Suppose that in Protocol PI2, a node $s \in V$ receives START. Then:

a) *all nodes $i \in V$ will accept the information in finite time and exactly once; after this happens, the links $\{(i, p_i)\ ,\ \forall i \in V\}$ will form a directed spanning tree rooted at s; in addition, for all i holds $t(phase1()_i) > t(phase1()_{p_i})$.*

b) *During the execution of the protocol, exactly one MSG is sent on each link of the type $\neq (i, p_i)$, in each direction. On links of the type (i, p_i), a MSG is sent only in the direction from p_i to i.*

c) *The propagation of information is the fastest possible.*

Properties of *PI2*

Theorem

Suppose that in Protocol PI2, a node $s \in V$ receives START. Then:

a) *all nodes $i \in V$ will accept the information in finite time and exactly once; after this happens, the links $\{(i, p_i)\ ,\ \forall i \in V\}$ will form a directed spanning tree rooted at s; in addition, for all i holds $t(phase1()_i) > t(phase1()_{p_i})$.*

b) *During the execution of the protocol, exactly one MSG is sent on each link of the type $\neq (i, p_i)$, in each direction. On links of the type (i, p_i), a MSG is sent only in the direction from p_i to i.*

c) *The propagation of information is the fastest possible.*

d) *No string of messages can overtake PI2.*

PI3

Sometimes we want to return the network to its initial state.

Protocol PI3

Messages

$MSG(info)$ - message carrying the information *info* to be propagated

Variables

G_i - set of neighbors of i

m_i - shows whether node i is in the protocol (values 0,1).

$e_i(l)$ - number of MSG's sent to neighbor l - number of MSG's received from it, for all $l \in G_i$

Initialization

if i receives a MSG, then

- just before receiving the first MSG, holds $m_i = 0$ and $e_i(l) = 0$ for all $l \in G_i$
- after receiving the first MSG and until m_i returns next to 0, node i discards and disregards messages not sent in the present instance of the protocol

Algorithm for node i

```
A1    receives MSG(info) from l ∈ G_i ∪ {nil}
A2    {   if  (m_i = 0) {
A3              phase1();
          }
A4        e_i(l) ← e_i(l) − 1;
A5        if  (e_i(k) = 0 ∀k ∈ G_i) phase2();
      }
B1    phase1()                                   /* similar to PI1 */
B2    {   m_i ← 1;
B3        accept(info);
B4        for  (k ∈ G_i){
B5            send MSG(info) to k;
B6            e_i(k) ← e_i(k) + 1;
          }
      }
C1    phase2()
C2    {   m_i ← 0;
      }
```

Note: recall that if *MSG* is received from *nil*, the lines containing $e_i(l)$ are disregarded.

Properties of *PI3*

Theorem

Suppose that in Protocol $PI3$, node $s \in V$ receives $START$. Then:

Properties of *PI3*

Theorem

Suppose that in Protocol PI3, node $s \in V$ receives $START$. Then:

a) *All nodes $i \in V$ will accept the information in finite time and exactly once.*

Properties of *PI3*

Theorem

Suppose that in Protocol $PI3$, *node* $s \in V$ *receives* $START$. *Then:*

a) *All nodes* $i \in V$ *will accept the information in finite time and exactly once.*
b) *During the execution of the protocol, exactly one* MSG *is sent on each link in each direction.*

Properties of *PI3*

Theorem

Suppose that in Protocol PI3, node $s \in V$ receives $START$. Then:

a) *All nodes $i \in V$ will accept the information in finite time and exactly once.*
b) *During the execution of the protocol, exactly one MSG is sent on each link in each direction.*
c) *The propagation of information is the fastest possible.*

Properties of *PI3*

Theorem

Suppose that in Protocol $PI3$, node $s \in V$ receives $START$. Then:

a) *All nodes $i \in V$ will accept the information in finite time and exactly once.*
b) *During the execution of the protocol, exactly one MSG is sent on each link in each direction.*
c) *The propagation of information is the fastest possible.*
d) *No string of messages can overtake $PI3$.*

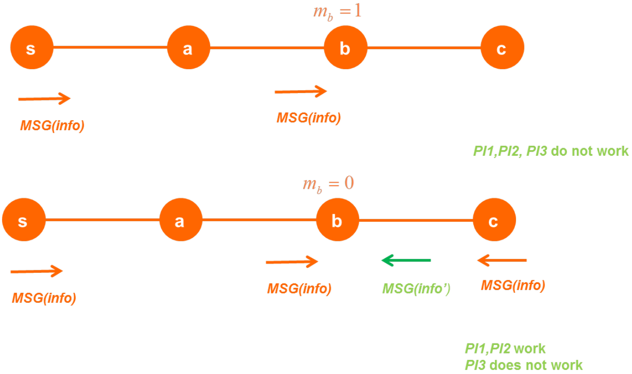
Properties of *PI3*

Theorem

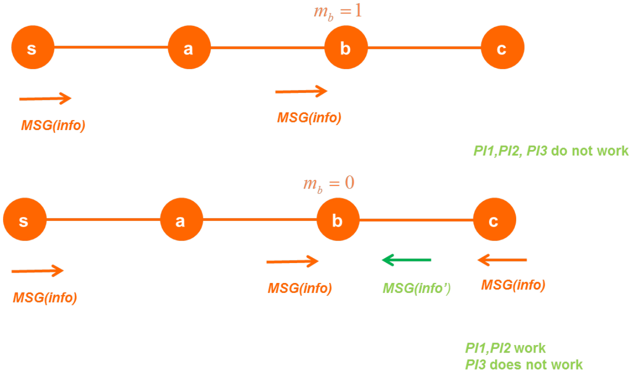
Suppose that in Protocol $PI3$, node $s \in V$ receives $START$. Then:

a) *All nodes $i \in V$ will accept the information in finite time and exactly once.*
b) *During the execution of the protocol, exactly one MSG is sent on each link in each direction.*
c) *The propagation of information is the fastest possible.*
d) *No string of messages can overtake $PI3$.*
e) *Every node $i \in V$ executes $phase2()_i$ in finite time and after this time it receives no more MSG's.*

Initial Conditions (Go to Animations.pptx)



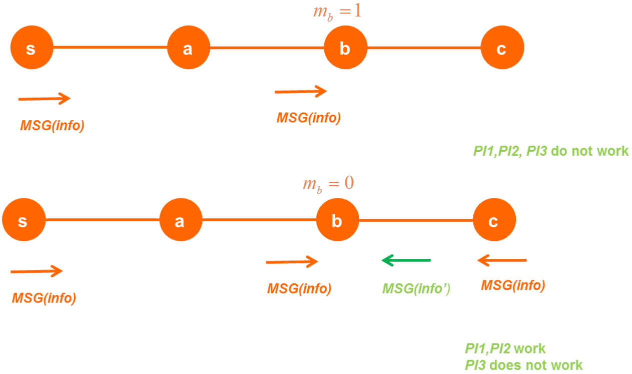
Initial Conditions (Go to Animations.pptx)



Why not simply use the "synchronous" Initial Conditions that most papers use: There is an initial time t_0 when:

- all nodes have $m_i = 0$ and
- there are no messages in the network

Initial Conditions (Go to Animations.pptx)



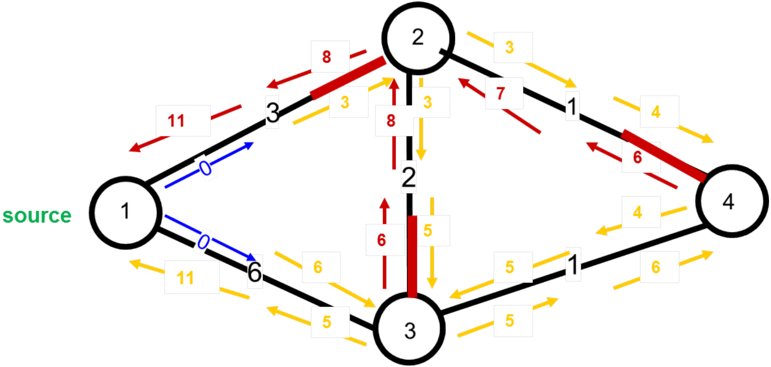
Why not simply use the "synchronous" Initial Conditions that most papers use: There is an initial time t_0 when:

- all nodes have $m_i = 0$ and
- there are no messages in the network

Answer: Too strong, these conditions do not hold when we'll need to use these Basic Protocols as a basis for more complicated ones (see Slide 31).

Propagation of Information with Feedback (PIF)

Example PIF



Go to Animation.pptx

PIF1

Protocol PIF1

Messages

$MSG(info)$ - message carrying the information *info* to be propagated

Variables

G_i - set of neighbors of i

m_i - shows if node i has already entered the protocol (values 0,1).

$e_i(l)$ - number of MSG's sent to l - number of MSG's received from l, for all $l \in G_i$

p_i - neighbor from which the first MSG is received

Initialization

if i receives a MSG, then

- just before receiving the first MSG, holds $m_i = 0$ and $e_i(k) = 0$ for all $k \in G_i$
- after receiving the first MSG, node i discards and disregards messages not sent in the present instance of the protocol

Note: By definition, a condition on an empty set is always true. For instance, in <$A5$> below, if $G_i - \{p_i\} = \emptyset$, then the condition holds and i should perform $phase2()$.

Algorithm for node i

A1 receives $MSG(info)$ from $l \in G_i \cup \{nil\}$
A2 $\{$ *if* $(m_i = 0)$ $\{$
A3 $\quad phase1();$
$\}$
A4 $e_i(l) \leftarrow e_i(l) - 1;$
A5 *if* $(e_i(k) = 0\ \forall k \in G_i - \{p_i\})$ $phase2();$
$\}$
B1 $phase1()$ /* similar to PI2 */
B2 $\{$ $m_i \leftarrow 1;$
B3 $p_i \leftarrow l;$
B4 accept($info$);
B5 *for* $(k \in G_i - \{p_i\})\{$
B6 $\quad$ send $MSG(info)$ to k;
B7 $\quad e_i(k) \leftarrow e_i(k) + 1;$
$\}$
$\}$
C1 $phase2()$
C2 $\{$ send $MSG(info)$ to p_i
C3 $e_i(p_i) \leftarrow e_i(p_i) + 1;$
$\}$

Note: recall that for a node i that receives MSG from nil, the parameter p_i becomes nil, the lines containing $e_i(l)$ are disregarded and when eventually node i performs $<C1>$ it sends MSG to no one.

Properties of *PIF1*

Theorem

*Suppose that in Protocol PI*3*, node* $s \in V$ *receives* $START$*. Then:*

Properties of *PIF1*

Theorem

Suppose that in Protocol PI3, node $s \in V$ receives START. Then:

a) *All nodes $i \in V$ will accept the information in finite time and exactly once.*

Properties of *PIF1*

Theorem

Suppose that in Protocol PI3, node $s \in V$ receives $START$. Then:

a) *All nodes $i \in V$ will accept the information in finite time and exactly once.*
b) *During the execution of the protocol, exactly one MSG is sent on each link in each direction.*

Properties of *PIF1*

Theorem

Suppose that in Protocol PI3, node $s \in V$ receives $START$. Then:

a) *All nodes $i \in V$ will accept the information in finite time and exactly once.*
b) *During the execution of the protocol, exactly one MSG is sent on each link in each direction.*
c) *The propagation of information is the fastest possible.*

Properties of *PIF1*

Theorem

Suppose that in Protocol PI3, node $s \in V$ receives START. Then:

a) *All nodes $i \in V$ will accept the information in finite time and exactly once.*
b) *During the execution of the protocol, exactly one MSG is sent on each link in each direction.*
c) *The propagation of information is the fastest possible.*
d) *No string of messages can overtake PI3.*

Properties of *PIF1*

Theorem

Suppose that in Protocol $PI3$, node $s \in V$ receives $START$. Then:

a) *All nodes $i \in V$ will accept the information in finite time and exactly once.*
b) *During the execution of the protocol, exactly one MSG is sent on each link in each direction.*
c) *The propagation of information is the fastest possible.*
d) *No string of messages can overtake $PI3$.*
e) *Every node $i \in V$ executes $phase2()_i$ in finite time and after this time it receives no more MSG's.*

Connectivity Test Protocols

The purpose of this class of DNP's is to allow each node to learn what nodes are connected to it, i.e. nodes that are in V.

Protocol CT1

Messages

MSG^j - control messages with identity j

Variables

G_i - set of neighbors of node i

m_i - shows whether i has already entered the algorithm (values 0,1)

c_i^j - designates knowledge at i about connectivity to j (values 0,1), for all $j \in \overline{V}$

Initialization

if a node receives at least one MSG,

- just before the time it receives the first one holds $m_i = 0$
- after receiving the first MSG, node i discards and disregards messages not sent in the present instance of the protocol

Algorithm for node i

A1 receives MSG^j from $l \in G_i \cup \{nil\}$
A2 $\{$ *if* $(m_i = 0)\{$
A3 $m_i \leftarrow 1;$ /* enter protocol */
A4 $initialize();$
A5 $phase1^i();$
$\}$
A6 *if* $(c_i^j = 0)\ phase1^j();$
$\}$
B1 $phase1^j()$
B2 $\{$ $c_i^j \leftarrow 1;$
B3 *for* $(k \in G_i)$ send MSG^j to $k;$
$\}$
C1 $initialize()$
C2 $\{$ *for* $(k \in \overline{V})\ c_i^k \leftarrow 0;$
$\}$

Properties of *CT1*

Theorem

Suppose that at least one node in V receives $START$. Then for every $i \in V$, the variables c_i^j will become 1 in finite time for all $j \in V$ and will remain 0 forever for all $j \notin V$.

Properties of *CT1*

Theorem

Suppose that at least one node in V receives $START$. Then for every $i \in V$, the variables c_i^j will become 1 in finite time for all $j \in V$ and will remain 0 forever for all $j \notin V$.

Theorem

With protocol $CT1$, there is no way for node j to know for sure what nodes are disconnected from it or in other words, there is no way for j to know when the algorithm is completed, except for the case when $V \equiv \overline{V}$.

CT2

The protocol is started and entered by nodes in the same way as in CT1, except that when it enters the protocol, every node j triggers a $PIF1^j$ with its identity j instead of a $PI1^j$ as in CT1. It is shown in the Theorem below that at the time it completes its own $PIF1$, a node j has complete knowledge about the identities of nodes in V and those that are not in V. Consequently, the termination property holds for Protocol CT2.

CT2

The protocol is started and entered by nodes in the same way as in CT1, except that when it enters the protocol, every node j triggers a $PIF1^j$ with its identity j instead of a $PI1^j$ as in CT1. It is shown in the Theorem below that at the time it completes its own $PIF1$, a node j has complete knowledge about the identities of nodes in V and those that are not in V. Consequently, the termination property holds for Protocol CT2.

Protocol CT2

Messages

MSG^j - control messages with identity j sent by i

Variables

G_i - set of neighbors of node i
m_i - indicates whether i has entered the protocol (values 0,1)
c_i^j - designates knowledge at i about connectivity to j (values 0,1) for all $j \in \overline{V}$
p_i^j - neighbor from which MSG^j has been received first, for all $j \neq i$.
$e_i^j(l)$ - number of MSG^j sent to l - number of MSG^j received from l, for all $l \in G_i$

Initialization

if a node receives at least one MSG, then

- just before the time it receives the first one, holds $m_i = 0$
- after receiving the first MSG, node i discards and disregards messages not sent in the present instance of the protocol

Algorithm for node i

A1 receives MSG^j from $l \in G_i \cup \{nil\}$
A2 $\{$ *if* $(m_i = 0)\{$
A3 $\quad m_i \leftarrow 1;$ /* enter protocol */
A4 $\quad$ *initialize*();
A5 $\quad$ *phase*1^i();
$\}$
A6 *if* $(c_i^j = 0)$ *phase*1^j();
A7 $e_i^j(l) \leftarrow e_i^j(l) - 1$
A8 *if* $(e_i^j(k) = 0 \; \forall k \in G_i - \{p_i^j\})$ *phase*2^j();
$\}$
B1 *phase*1^j() /* same as PIF1 */
B2 $\{$ $c_i^j \leftarrow 1;$
B3 *if* $(i \neq j)$ $p_i^j \leftarrow l$ *else* $p_i^j \leftarrow nil;$
B4 *for* $(k \in G_i - \{p_i^j\})\{$
B5 $\quad$ send MSG^j to k;
B6 $\quad e_i^j(k) \leftarrow e_i^j(k) + 1;$
$\}$
$\}$
C1 *phase*2^j() /* same as PIF1 */
C2 $\{$ send MSG^j to p_i^j
C3 $e_i^j(p_i^j) \leftarrow e_i^j(p_i^j) + 1;$
$\}$
D1 *initialize*()
D2 $\{$ *for* $(j \in \overline{V})\{$
D3 $\quad c_i^j \leftarrow 0;$
D4 $\quad$ *for* $(k \in G_i)$ $e_i^j(k) \leftarrow 0;$
$\}$
$\}$

Properties of *CT2*

Theorem

Suppose that at least one node in V receives $START$. Then:

Properties of *CT2*

Theorem

Suppose that at least one node in V receives $START$. Then:

a) *at every node $i \in V$, the variables c_i^j will become 1 in finite time for all $j \in V$ and will remain 0 forever for all $j \notin V$.*

Properties of *CT2*

Theorem

Suppose that at least one node in V receives $START$. Then:

a) *at every node $i \in V$, the variables c_i^j will become 1 in finite time for all $j \in V$ and will remain 0 forever for all $j \notin V$.*

b) *every $i \in V$ will perform $phase2()_i^i$ in finite time and exactly once, and when this happens, it will have $c_i^j = 1$ for all $j \in V$ and $c_i^j = 0$ for all $j \notin V$. In other words, it will positively know at that time what nodes are connected.*

CT3

The protocol is the same as CT1, except that for every node j , $PI1^j$ propagates also the neighbors of j. Nodes will know when the neighbors of all nodes have been accounted for.

CT3

The protocol is the same as CT1, except that for every node j , $PI1^j$ propagates also the neighbors of j. Nodes will know when the neighbors of all nodes have been accounted for.

Protocol CT3

Messages

$MSG^j(\Lambda)$ - control messages with identity j and $\Lambda = G_j$

Variables

G_i - set of neighbors of node i
m_i - shows whether i has already entered the algorithm (values 0,1)
c_i^j - designates knowledge at i about connectivity to j (values 0,1,2), for all $j \in \overline{V}$
$= 0$ when i knows nothing about j
$= 1$ while i knows j only as a neighbor of another node
$= 2$ while i knows j directly (i.e. $MSG^j(\Lambda)$ has been received)

Initialization

if a node receives at least one MSG, then

- just before the time it receives the first one holds $m_i = 0$
- after receiving the first MSG, node i discards and disregards messages not sent in the present instance of the protocol

Algorithm for node i

A1 receives $MSG^j(\Lambda)$ from $l \in G_i \cup \{nil\}$

A2 { *if* $(m_i = 0)\{$

A3 $m_i \leftarrow 1;$ /* enter protocol */

A4 $initialize();$

A5 $phase1^i(G_i);$

}

A6 *if* $(c_i^j \neq 2)$ $phase1^j(\Lambda);$

A7 *if* $(c_i^j = 0$ or $2, \forall j \in \overline{V})$ connectivity known;

}

B1 $phase1^j(\Lambda)$

B2 { $c_i^j \leftarrow 2;$

B3 *for* $(k \in \Lambda)$ $c_i^k \leftarrow max(c_i^k, 1);$

B4 *for* $(k \in G_i)$ send $MSG^j(\Lambda)$ to k;

}

C1 $initialize()$

C2 { *for* $(j' \in \overline{V})$ $c_i^{j'} \leftarrow 0;$

}

Properties of *CT3*

Theorem

Suppose that at least one node in V receives $START$. Then:

Properties of *CT3*

Theorem

Suppose that at least one node in V receives $START$. Then:

a) *for every $i \in V$, the variables c_i^j will become 2 in finite time for all $j \in V$ and will remain 0 forever for all $j \notin V$.*

Properties of *CT3*

Theorem

Suppose that at least one node in V receives $START$. Then:

a) *for every $i \in V$, the variables c_i^j will become 2 in finite time for all $j \in V$ and will remain 0 forever for all $j \notin V$.*

b *every $i \in V$ will perform $<A7>_i$ in finite time, and when this happens for the first time, it will have $c_i^j = 2$ for all $j \in V$ and $c_i^j = 0$ for all $j \notin V$. In other words, it will positively know at that time what nodes are connected.*

Extending CT to changing topologies - sequence numbers - ECT3

The CT Protocols require specific initial conditions and therefore their extension to handle topological changes must include re-initialization after every such change. This can be implemented by restarting a new cycle of the protocol after every topological event. In order to distinguish between messages and node states belonging to different cycles, we employ global sequence numbers.

Extending CT to changing topologies - sequence numbers - ECT3

The CT Protocols require specific initial conditions and therefore their extension to handle topological changes must include re-initialization after every such change. This can be implemented by restarting a new cycle of the protocol after every topological event. In order to distinguish between messages and node states belonging to different cycles, we employ global sequence numbers.

Protocol ECT3

Messages

$MSG^j(R, \Lambda)$ - control messages with identity j and $\Lambda =$ list G_j of neighbors of j

Variables

G_i - set of neighbors of node i

c_i^j - designates knowledge at i about connectivity to j (values 0,1,2), for all $j \in \overline{V}$

$= 0$ when i knows nothing about j

$= 1$ while i knows j only as a neighbor of another node

$= 2$ while i knows j directly (i.e. $MSG^j(\Lambda)$ has been received)

R_i - highest sequence number known to i (values: 0,1, ...);

Algorithm for node i

A1 node i becomes operational
A2 $\{ \quad R_i \leftarrow 0;$
$\}$
B1 link (i, l) enters *Connected* state or *Initialization* Mode
B2 $\{ \quad$ update G_i;
B3 $\quad R_i \leftarrow R_i + 1;$ /* enter protocol, replaces $m_i \leftarrow 1$ */
B4 $\quad$ *initialize*();
B5 $\quad phase1^i(G_i);$
$\}$
C1 receives $MSG^j(R, \Lambda)$ from $l \in G_i$
C2 $\{ \quad$ *if* $(R \geq R_i)\{$
C3 $\quad\quad$ *if* $(R > R_i)\{$
C4 $\quad\quad\quad R_i \leftarrow R;$ /* enter protocol, replaces $m_i \leftarrow 1$*/
C5 $\quad\quad\quad$ *initialize*();
C6 $\quad\quad\quad phase1^i(G_i);$
$\quad\quad \}$
C7 $\quad\quad$ *if* $(c_i^j \neq 2)$ $phase1^j(\Lambda);$
C8 $\quad\quad$ *if* $(c_i^j = 0$ or $2, \forall j \in \overline{V})$ connectivity known;
$\quad \}$
$\}$
D1 $phase1^j(\Lambda)$
D2 $\{ \quad c_i^j \leftarrow 2;$
D3 $\quad$ *for* $(k \in \Lambda)$ $c_i^k \leftarrow max(c_i^k, 1);$
D4 $\quad$ *for* $(k \in G_i)$ send $MSG^j(R_i, \Lambda)$ to k;
$\}$
E1 *initialize*()
E2 $\{ \quad$ *for* $(j \in \overline{V})$ $c_i^j \leftarrow 0;$
$\}$

Note that <*B*3> and <*C*4> here correspond to <*A*3> in CT3.

Properties of *ECT3*

Theorem

Consider an arbitrary finite *sequence of topological events with arbitrary timing and location and let (E, V) denote a connected subnetwork in the final topology within each at least one node has entered the protocol. Then there is a finite time after the sequence is completed after which no messages travel in (V, E) and all nodes $i \in V$ will have the same cycle number R_i, with $c_i^k = 2$ for all $k \in V$ and with $c_i^k = 0$ for all $k \notin V$.*

Proof.

Consider the topology of the network after all topological changes cease. Consider in this topology a given connected subnetwork (V, E). From <B3>, each topological event adjacent to a node $i \in V$ increments the cycle counter R_i at node i. Let $\{i_n\}$ be the collection of nodes in V that register change of status of an adjacent link, and let $\{t_n\}$ be the corresponding collection of times when the status change is registered. Since there is a *finite number of topological events*, the collections $\{i_n\}$, $\{t_n\}$ are **finite**. Let $R = \max\{R_{i_n}(t_n+)\}$ over all n. Then:

- R is the *highest* cycle number ever known in network (V, E)
- The cycle with number R is started by (one or more) nodes $i \in \{i_n\} \in V$ that increment their R_i to R as a result of a topological event. These nodes can be considered as if they receive START in CT3 and, indeed, the network covered by the cycle with number R registers no more topological events, since no counter number R_i is ever increased to $(R + 1)$.
- The initial conditions of CT3 hold for the R cycle as follows:
  - A node i is considered as having $m_i = 0$ or $m_i = 1$, depending if with $R_i < R$ or $R_i = R$.
  - Since R_i is nondecreasing, the first $MSG(R)$ that arrives at a node i finds $R_i < R$, i.e. $m_i = 0$.
  - After $R_i \leftarrow R$, a node disregards all messages with sequence number less than R, so that the condition that nodes receive only messages of the present protocol is also satisfied
- Moreover, from the Follow-up property of DLC follows that in the final topology, $l \in G_i$ if and only if $i \in G_l$, so that the assumption of bi-directionality holds in the final topology.

Consequently, the evolution of the cycle with sequence number R is the same as in protocol CT3 on (V, E), completing the proof.

□

Initial Conditions

Here we can see for the first time the reason for requiring asynchronous Initial Conditions in the Fixed Topology algorithms as opposed to synchronous ones: "there is a time t_0 when all $m_i = 0$ and there are no messages on the links". One can attempt to find such a time t_0 for example the time when the first message with $R_i = R$ is received by any node in (V, E). However, there is no guarantee that at that time there are no messages on the links. Some links may even have messages with $R_i = R$.