Epistemic Protocols for Distributed Gossiping

Krzysztof R. Apt

CWI, Amsterdam

Based on joint work with

Wiebe van der Hoek

and

Davide Grossi

“Your slot is divided into 20 minutes for your talk and 5 minutes for possible discussions and for switching the speaker.”
Pope Francis has sharply criticised the Vatican bureaucracy in a pre-Christmas address to cardinals, complaining of “spiritual Alzheimer’s” and “the terrorism of gossip”.

BBC News, 22 December 2014
Gossip Protocols

Example

- $n$ people, each knows a secret.
- How many phone calls are necessary before everybody knows every secret?
- In each call all secrets are exchanged.

**Theorem** (many authors, early seventies)
Assume $n \geq 4$.
At least $2n - 4$ calls are needed.
A solution with 4 calls for $n = 4$

A call: $(a, b)$.

- $A = \{a, b, c, d\}$.
- Take the sequence
  $(a, b), (c, d), (a, d), (b, c)$.

- After it $a, b, c, d$ know all the secrets.
A solution with 4 calls for $n = 4$

A call: $(a, b)$.

- $A = \{a, b, c, d\}$.
- Take the sequence
  $(a, b), (c, d), (a, d), (b, c)$.

- After it $a, b, c, d$ know all the secrets.
- Note that the sequence
  $(a, b), (b, c), (c, d), (d, a)$
  does not do the job.
A solution with $2n - 4$ calls for $n \geq 4$

- Let $A = \{a, b, c, d, i_1, \ldots, i_{n-4}\}$.
- Take the sequence
  $(a, i_1), (a, i_2), \ldots, (a, i_{n-4})$.
  After it $a$ knows all the secrets of $i_1, \ldots, i_{n-4}$.
- Follow it by the sequence
  $(a, b), (c, d), (a, d), (b, c)$.
  After it $a, b, c, d$ know all the secrets.
- Follow it by the sequence
  $(a, i_1), (a, i_2), \ldots, (a, i_{n-4})$.
  After it all the agents know all the secrets.

**Note** This is a centralized algorithm.
Gossip Algorithms

- A vast area.

Krzysztof R. Apt
Epistemic Protocols for Distributed Gossiping
Gossip Algorithms

- A vast area.
- (Hedetniemi, Hedetniemi and Liestman, '88)
  A survey of gossiping and broadcasting in communication networks. It has 135 references.
Gossip Algorithms

- A vast area.
- (Hedetniemi, Hedetniemi and Liestman, ’88) *A survey of gossiping and broadcasting in communication networks.* It has 135 references.
- **Distributed gossip protocols:** each agent acts autonomously, by passing on the gossips he knows and/or by soliciting gossips he does not know.
Distributed Gossip Protocols based on Epistemic Logic

- Introduced in Attamah, Van Ditmarsch, Grossi and Van der Hoek, ’14.
- Written as formulas of Dynamic Epistemic Logic.
- Assumptions:
  - A finite set of agents.
  - Each agent holds a (private) secret.
  - Each secret is viewed a distinct propositional variable.
  - Secret of agent \( a \) denoted by \( A \).
Our Setup (differences w.r.t. [ADGH14])

- Modes of Communication
  - push, \(a \triangleright b\),
  - pull, \(a \triangleleft b\),
  - push-pull, \(ab\) or \((a, b)\).

- In [ADGH14]) only push-pull was considered.
- No agent \(c \neq a, b\) notes that a call between \(a\) and \(b\) takes place.
- Programs written in the CSP style.
- Fairness considered (here omitted).
Epistemic formulas

\[ \phi ::= F_a p \mid \neg \phi \mid \phi \land \phi \mid K_a \phi, \]

where \( a \) is an agent and \( p \) a secret.

Intuition.
\( F_a p \) is a primitive formula: agent \( a \) is familiar with (knows) secret \( p \).
Example: \( F_a B \).

Intuition.
\( K_a \phi \) is not a primitive formula: agent \( a \) knows formula \( \phi \).
Example: \( K_a F_b C \).
Component program for an agent $a$:

$$\ast \left( \left[ \[] \right]_{j=1}^{m} \phi_j \rightarrow S_j \right),$$

where

- each $\phi_j$ is an epistemic formula (a guard),
- each $S_j$ is a call in which agent $a$ is the caller.

Distributed protocol: a parallel composition of component programs.
Example Protocol

- Consider the agents 1, 2, . . . , n arranged in a ring, where n ≥ 3.

- Program for agent $i$:

  $\neg K_i F_{i \oplus 1} I \ominus 1 \rightarrow (i, i \oplus 1)$.

- Agent $i$ calls his successor, $i \oplus 1$, if $i$ does not know whether his successor is familiar with the secret of $i$’s predecessor, $i \ominus 1$. 
Example Protocol, ctd

- Program for agent $i$:

\[ [\neg K_i F_{i \oplus 1} I \oplus 1 \rightarrow (i, i \oplus 1)]. \]
Example Protocol, ctd

- Program for agent $i$:

$$\ast [\neg K_i F;_{i \oplus 1} I \ominus 1 \rightarrow (i, i \oplus 1)].$$

- This protocol is incorrect.
Example Protocol, ctd

- Program for agent $i$:

\[
* [\neg K_i F_{i \oplus 1} I \ominus 1 \rightarrow (i, i \oplus 1)].
\]

- This protocol is incorrect.

- The sequence of calls

\[
ab, bc, cd, de, ea, ab
\]

results in a termination.
Example Protocol, ctd

- Program for agent $i$:

\[\neg K_i F_{i \oplus 1} I \oplus 1 \rightarrow (i, i \oplus 1)\].

- This protocol is incorrect.

- The sequence of calls

\[ab, bc, cd, de, ea, ab\]

results in a termination.

- However, at this point agent $c$ does not know the secret of agent $e$. 
Computations

- Semantics of a gossip protocol: a computation tree.
Computations

- Semantics of a gossip protocol: a computation tree.
- The nodes are call sequences.
Semantics of a gossip protocol: a computation tree.

The nodes are call sequences.

Each call extends the current call sequence.
Computation

- Semantics of a gossip protocol: a computation tree. 
- The nodes are call sequences.
- Each call extends the current call sequence.
- This requires a semantics to evaluate epistemic formulas (the guards) (a formula is true in a call sequence).
Computations

- Semantics of a gossip protocol: a computation tree.
- The nodes are call sequences.
- Each call extends the current call sequence.
- This requires a semantics to evaluate epistemic formulas (the guards) (a formula is true in a call sequence).
- A computation of a gossip protocol: a maximal rooted path in its computation tree.
Assume a gossip protocol $P$ for the set of agents $A$.

- $P$ is partially correct if
  \[ \bigwedge_{a,b \in A} F_a B \]
  is true in all leaves of the computation tree of $P$.

- $P$ terminates if all its computations are finite.
Another Ring Protocol

● Program for agent $i$:

\[
\ast \left[ (\neg \bigwedge_{j=1}^{n} F_{i,j}) \lor \neg K_{i} F_{i \oplus 1} I \ominus 1 \rightarrow (i, i \ominus 1) \right].
\]

● Agent $i$ calls his successor, $i \oplus 1$, if he is not familiar with all the secrets or he does not know whether his successor is familiar with the secret of $i$’s predecessor, $i \ominus 1$.

● This protocol is partially correct.

● Termination:

<table>
<thead>
<tr>
<th>T</th>
<th>T for $\triangleright$</th>
<th>T for $\triangleleft$</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
Another Ring Protocol

- Program for agent $i$:

$$\ast[(\neg \bigwedge_{j=1}^{n} F_{i,j}) \lor \neg K_i F_{i\oplus 1} \to (i, i \oplus 1)]$$

- Agent $i$ calls his successor, $i \oplus 1$, if he is not familiar with all the secrets or he does not know whether his successor is familiar with the secret of $i$’s predecessor, $i \ominus 1$.

- This protocol is partially correct.

- Termination:

<table>
<thead>
<tr>
<th></th>
<th>T for $\triangleright$</th>
<th>T for $\triangleright$</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

- Non-termination for push-pull: $(i, i \ominus 1), (i, i \oplus 1), \ldots$
A Terminating Ring Protocol

- Program for agent $i$:

\[
\bigvee_{j=1}^{n} (F_i J \land \neg K_i F_{i \oplus 1} J) \rightarrow (i, i \oplus 1).
\]

- Agent $i$ calls his successor, $i \oplus 1$, if $i$ is familiar with some secret and he does not know whether his successor is familiar with it.

- Termination:

<table>
<thead>
<tr>
<th></th>
<th>T for $\rhd$</th>
<th>T for $\rhd$</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
A Terminating Ring Protocol

- Program for agent $i$:

$$\forall j=1^n \left[ (F_i \land \neg K_i F_{i \oplus 1} J) \rightarrow (i, i \oplus 1) \right].$$

- Agent $i$ calls his successor, $i \oplus 1$, if $i$ is familiar with some secret and he does not know whether his successor is familiar with it.

- Termination:

<table>
<thead>
<tr>
<th></th>
<th>T for $\triangleright$</th>
<th>T for $\triangleleft$</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

- Non-termination for $\triangleleft$: $i \triangleleft i \oplus 1, i \triangleleft i \oplus 1, \ldots$
Protocols for Complete Graphs

Learn New Secrets Protocol ([ADGH14])

- Program for agent $i$:

$$*[[[]_{j \in A} \neg F_i J \rightarrow (i, j)].$$

- Agent $i$ calls agent $j$ if agent $i$ is not familiar with his secret.

- Termination:

<table>
<thead>
<tr>
<th>T</th>
<th>T for ⊳</th>
<th>T for ⊲</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
Protocols for Complete Graphs

Learn New Secrets Protocol ([ADGH14])

- Program for agent $i$:

  \[ *[[j \in A \neg F_i J \rightarrow (i, j)]. \]

  
- Agent $i$ calls agent $j$ if agent $i$ is not familiar with his secret.

- Termination:

<table>
<thead>
<tr>
<th>T</th>
<th>T for $\triangleright$</th>
<th>T for $\triangleright$</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

  
- Non-termination for $\triangleright$: $a \triangleright b$, $a \triangleright b$, ...
Learn New Secrets Protocol for push-pull

- Program for agent $i$:

  $*[[j \in A \rightarrow F_i J] \rightarrow (i, j)].$

- It can generate the shortest sequences of $2n - 4$ calls.
- It can also generate the longest sequences of $\frac{n(n-1)}{2}$ calls.
Hear My Secret Protocol

- Program for agent $i$:

  $$\ast[[[]_{j \in A} \neg KiFjI \rightarrow (i, j)].$$

- Agent $i$ calls agent $j$ if agent $i$ does not know whether $j$ is familiar with his secret.

- Termination:

<table>
<thead>
<tr>
<th></th>
<th>T for $\triangleright$</th>
<th>T for $\triangleleft$</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

Krzysztof R. Apt

Epistemic Protocols for Distributed Gossiping
Hear My Secret Protocol

- Program for agent $i$:

  \[ \star [\vDash_{j \in A} \neg K_i F_j I \rightarrow (i,j)] \, . \]

- Agent $i$ calls agent $j$ if agent $i$ does not know whether $j$ is familiar with his secret.

- Termination:

<table>
<thead>
<tr>
<th>T for $\triangleright$</th>
<th>T for $\triangleleft$</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

- Non-termination for $\triangleleft$:
  Suppose everybody repeatedly calls $i$. Then $i$ never learns anything.
Hear My Secret Protocol

- Program for agent $i$:

$$\star [[j \in A \neg K_i F_j I \rightarrow (i, j)].$$

- Agent $i$ calls agent $j$ if agent $i$ does not know whether $j$ is familiar with his secret.

- Termination:

<table>
<thead>
<tr>
<th></th>
<th>T for $\triangleright$</th>
<th>T for $\triangleleft$</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

- Non-termination for $\triangleleft$:
  Suppose everybody repeatedly calls $i$. Then $i$ never learns anything.

- It can generate the shortest and the longest sequences of calls.
Thank you Ernst
Thank you Ernst
for many years of fruitful cooperation
Thank you Ernst
for many years of fruitful cooperation
and for your friendship