Epistemic Gossip Protocols

Krzysztof R. Apt

CWI

Based on joint work with

Wiebe van der Hoek

and

Davide Grossi
Anyone who has obeyed nature by transmitting a piece of gossip experiences the explosive relief that accompanies the satisfying of a primary need.

Primo Levi
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)
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.
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 it knows and/or by soliciting gossips it does not know.
Distributed Gossip Protocols based on Epistemic Logic

Assumptions (Attamah, Van Ditmarsch, Grossi and Van der Hoek, ’14).

- **Agents and secrets.**
  - A finite set \( A \) of at least two agents.
  - Each agent holds a secret.
  - Each secret is viewed a distinct propositional variable.

- **Types of calls.**
  - \( ab^- \): every agent \( c \neq a, b \) noted that \( a \) called \( b \),
  - \( ab^0 \): every agent \( c \neq a, b \) noted that some call took place, though not between whom,
  - \( ab^+ \): every agent \( c \neq a, b \) noted that possibly some call took place, though not between whom.

In all three cases no agent \( c \neq a, b \) learns the contents of the call.

- **Intuition.** The superscript indicates the degree of privacy of the call: \(- < 0 < +\).
Our Setup

- **Type of calls.**
  
  \( ab: \) no agent \( c \neq a, b \) noted that the call between \( a \) and \( b \) took place.

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

- In [ADGH14] only push-pull was considered.
Syntax

- **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 \( p \).
Syntax, ctd (modelled after CSP [Hoare '78])

Component program for an agent $a$:

$$*[[[]]_{j=1}^m \phi_j \rightarrow S_j],$$

where

- each $\phi_j$ is an epistemic formula,
- 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 \( n \geq 3 \).

Program for agent \( i \):

\[
\star [\neg K_i F_i \oplus 1 \ominus 1 \rightarrow (i, i \oplus 1)].
\]

- 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 $i$:

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

This protocol is not correct. 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 $e$. 
Example Protocol, ctd

- Program for agent $i$:

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

- This protocol is not correct.
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 not correct.

- The sequence of calls

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

  results in a termination.
Example Protocol, ctd

- Program for agent $i$:

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

- This protocol is not correct.

- 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 $e$.
Semantics (modification of [ADGH14])

- **Gossip situation**: \((Q_a)_{a \in A}\), where \(\forall a \ Q_a \subseteq P\).
  Intuition. \(Q_a\) is the set of secrets \(a\) knows.

- **Initial gossip situation** (root):
  each \(Q_a\) equals \(\{A\}\).
  Intuition. Initially each agent knows only his own secret.
Transformation of the Gossip Situations

- Each call transforms the current gossip situation by adjusting the relevant $Q_i$ relations.
Transformation of the Gossip Situations

- Each call transforms the current gossip situation by adjusting the relevant $Q_i$ relations.

- $ab(G) = G'$, where
  
  $Q'_a = Q'_b = Q_a \cup Q_b,$
  
  $Q'_c = Q_c$ for $c \neq a, b,$
Transformation of the Gossip Situations

- Each call transforms the current gossip situation by adjusting the relevant $Q_i$ relations.

  \[ ab(G) = G', \text{ where} \]
  \[ Q'_a = Q'_b = Q_a \cup Q_b, \]
  \[ Q'_c = Q_c \text{ for } c \neq a, b, \]

- \[ (a \diamond b)(G) = G', \text{ where} \]
  \[ Q'_b = Q_a \cup Q_b, \]
  \[ Q'_a = Q_a, \]
  \[ Q'_c = Q_c \text{ for } c \neq a, b, \]
Transformation of the Gossip Situations

- Each call transforms the current gossip situation by adjusting the relevant $Q_i$ relations.

- $ab(G) = G'$, where
  
  $Q'_a = Q'_b = Q_a \cup Q_b$,
  
  $Q'_c = Q_c$ for $c \neq a, b$,

- $(a \triangleright b)(G) = G'$, where
  
  $Q'_b = Q_a \cup Q_b$,
  
  $Q'_a = Q_a$,
  
  $Q'_c = Q_c$ for $c \neq a, b$,

- $(a \triangleleft b)(G) = G'$, where
  
  $Q'_a = Q_a \cup Q_b$,
  
  $Q'_b = Q_b$,
  
  $Q'_c = Q_c$ for $c \neq a, b$. 

Krzysztof R. Apt

Epistemic Gossip Protocols
Example

- Consider three agents, $a, b, c$ and the sequence of three calls $bc, ac, bc$.
- In the initial gossip situation root

$$Q_a = \{A\}, \quad Q_b = \{B\}, \quad Q_c = \{C\}.$$
Example

- Consider three agents, $a$, $b$, $c$ and the sequence of three calls $bc, ac, bc$.
- In the initial gossip situation root

$$Q_a = \{A\}, \quad Q_b = \{B\}, \quad Q_c = \{C\}.$$  

- The first call transforms it into $G_1$ in which

$$Q_a = \{A\}, \quad Q_b = \{B, C\}, \quad Q_c = \{B, C\}.$$
Example

- Consider three agents, \(a\), \(b\), \(c\) and the sequence of three calls \(bc, ac, bc\).
- In the initial gossip situation root

\[
Q_a = \{A\}, \quad Q_b = \{B\}, \quad Q_c = \{C\}.
\]

- The first call transforms it into \(G_1\) in which

\[
Q_a = \{A\}, \quad Q_b = \{B, C\}, \quad Q_c = \{B, C\}.
\]

- The second call transforms it into \(G_2\) in which

\[
Q_a = \{A, B, C\}, \quad Q_b = \{B, C\}, \quad Q_c = \{A, B, C\}.
\]
Example

- Consider three agents, $a, b, c$ and the sequence of three calls $bc, ac, bc$.
- In the initial gossip situation root

$$Q_a = \{A\}, \quad Q_b = \{B\}, \quad Q_c = \{C\}.$$  

- The first call transforms it into $G_1$ in which

$$Q_a = \{A\}, \quad Q_b = \{B, C\}, \quad Q_c = \{B, C\}.$$  

- The second call transforms it into $G_2$ in which

$$Q_a = \{A, B, C\}, \quad Q_b = \{B, C\}, \quad Q_c = \{A, B, C\}.$$  

- The third call transforms it into $G_3$ in which

$$Q_a = \{A, B, C\}, \quad Q_b = \{A, B, C\}, \quad Q_c = \{A, B, C\}.$$
Call Sequences

- $\vec{C}$: the set of all finite call sequences.
  - $\vec{c}(\text{root})$: the result of iteratively applying the calls in $\vec{c}$ to root.

- In the previous example:
  - $\vec{c} = (bc, ac, bc)$,
  - $\vec{c}(\text{root}) = (\{A, B, C\}, \{A, B, C\}, \{A, B, C\})$. 
Gossip models

- The **gossip model** captures the idea that agents are uncertain which call sequence took place.
- The gossip model: \((\vec{C}, (\sim_a)_{a \in A})\), where
  - for each \(a \in A\)
    \[ \vec{c} \sim_a \vec{c}' \text{ iff agent } a \text{ cannot distinguish between } \vec{c} \text{ and } \vec{c}'. \]
- A consequence of \(\vec{c} \sim_a \vec{c}'\):
  \[ \vec{c}(\text{root})_a = \vec{c}'(\text{root})_a. \]
Consider the gossip model $M := (\vec{C}, (\sim_a)_{a \in A})$ and $\vec{c} \in \vec{C}$.

Let $\vec{c}$(root) $= (Q_a)_{a \in A}$.

- $(M, \vec{c}) \models F_a p$ iff $p \in Q_a$,
- $(M, \vec{c}) \models K_a \phi$ iff $(M, \vec{c}') \models \phi$ for every $\vec{c}' \sim_a \vec{c}$.
Computation (1)

- Semantics of a gossip protocol: a computation tree.
- The nodes are call sequences.
- Each call extends the current call sequence.
Assume a gossip protocol that is a parallel composition of
\( [\emptyset]_{j=1}^{m_a} \phi_j^a \rightarrow S_j^a \), where \( a \in A \).

Then

\[
(c_0, \ldots, c_i) \rightarrow (c_0, \ldots, c_i, c_{i+1})
\]

if for some \( a \) and \( j \in \{1, \ldots, m_a\} \)

\[
(M, (c_0, \ldots, c_i)) \models \phi_j^a \text{ and } S_j^a = c_{i+1}.
\]
Assume a gossip protocol that is a parallel composition of
\( \ast \left[ \left[ \bigwedge_{j=1}^{m_a} \phi_j^a \rightarrow S_j^a \right] \right] \), where \( a \in A \).

Then

\[(c_0, \ldots, c_i) \rightarrow (c_0, \ldots, c_i, c_{i+1})\]

if for some \( a \) and \( j \in \{1, \ldots, m_a\} \)

\[(M, (c_0, \ldots, c_i)) \models \phi_j^a \text{ and } S_j^a = c_{i+1}.\]

So \( (c_0, \ldots, c_k) \) is a leaf of the computation tree iff

\[(M, (c_0, \ldots, c_k)) \models \bigwedge_{a \in A} \bigwedge_{j=1}^{m_a} \neg \phi_j^a.\]
Assume $\ast[[\phi_j^a \rightarrow S_j^a]]$, where $a \in A$.

A computation of a gossip protocol: a maximal rooted path in its computation tree.

An agent $a$ is enabled in $\vec{c}$ from $\xi$ if $(M, \vec{c}) \models \bigvee_{j=1}^{m_a} \phi_j^a$.

Intuition. An agent is enabled if it can perform a call.

An agent $a$ is selected in $\vec{c}$ from $\xi$ if it is the caller in the call that caused the transition $\vec{c} \rightarrow \vec{c'}$ in $\xi$.

A computation $\xi$ is fair if it is finite or each agent enabled in infinitely many situation sequences in $\xi$ is selected in infinitely many situation sequences in $\xi$. 
Correctness

Assume a gossip protocol $P$ that is a parallel composition of $\star[[[]_{j=1}^{m_a} \phi_j^a \rightarrow S_j^a]],$ where $a \in A.$

- $P$ 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.

- $P$ **fairly terminates** if all its fair computations are finite.
Some Implications

T: termination.
FT: fair termination.

- **Note**
  - \( T \implies FT \).
  - \( T \text{ for } △ \implies FT \text{ for } △ \).
  - \( T \text{ for } ◁ \implies FT \text{ for } ◁ \).
- No other implications hold.
Example: $T$ for $\triangleright$ does not imply $T$

- Consider three agents, $a, b, c$.
- $F_j\text{All}$ stands for $\wedge_{i \in \{a, b, c\}} F_j I$.
Example: T for $\triangleright$ does not imply T

- Consider three agents, $a, b, c$.
- $F_j All$ stands for $\bigwedge_{i \in \{a, b, c\}} F_j I$.
- Let $A \subseteq C$ stand for
  \[
  \bigwedge_{i \in \{a, b, c\}} (F_a I \rightarrow F_c I) \land \bigvee_{i \in \{a, b, c\}} (F_c I \land \neg F_a I)
  \]

  **Intuition.** $c$ is familiar with more secrets than $a$.

- Consider the following component programs:
  - for agent $a$: $\star [\neg (A \subseteq C) \land \neg F_a All \rightarrow a \triangleright c]$
  - for agent $b$: $\star [\neg (B \subseteq C) \land \neg F_b All \rightarrow b \triangleright c]$
  - for agent $c$: $\star [\square_{i \in \{a, b\}} F_c All \land \neg K_c F_i C \rightarrow c \triangleright i]$. 

This protocol is correct, terminates for $\triangleright$ and does not always terminate for the push-pull mode.

For the push-pull mode $ac$, $ac$, $ac$, ... is a possible computation.
Example: $T$ for $\triangleright$ does not imply $T$

- Consider three agents, $a, b, c$.
- $F_j All$ stands for $\bigwedge_{i \in \{a, b, c\}} F_j I$.
- Let $\mathcal{A} \subset \mathcal{C}$ stand for
  \[\bigwedge_{i \in \{a, b, c\}} (F_a I \rightarrow F_c I) \land \bigvee_{i \in \{a, b, c\}} (F_c I \land \neg F_a I)\]

- **Intuition.** $c$ is familiar with more secrets than $a$.
- Consider the following component programs:
  - for agent $a$: $*[\neg(\mathcal{A} \subset \mathcal{C}) \land \neg F_a All \rightarrow a \triangleright c]$,  
  - for agent $b$: $*[\neg(\mathcal{B} \subset \mathcal{C}) \land \neg F_b All \rightarrow b \triangleright c]$,  
  - for agent $c$: $*[\square[[]]_{i \in \{a, b\}} F_c All \land \neg K_c F_i C \rightarrow c \triangleright i]$.

- This protocol is correct, terminates for $\triangleright$ and does not always terminate for the push-pull mode.
- The only computations: $[a \triangleright c \parallel b \triangleright c], [c \triangleright a \parallel c \triangleright b]$.
- For the push-pull mode $ac, ac, ac, \ldots$ is a possible computation.
Another Ring Protocol

- **Program for agent** $i$:

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

where $\lozenge$ is $\triangleright$, $\triangleleft$ or push-pull.

- **Agent** $i$ calls his successor, $i \oplus 1$, if $i$ is familiar with some secret and he knows that his successor is not familiar with it.

- **Properties of this protocol:**
  
  - $\lozenge = \triangleright$.
    - It is correct and it terminates.
  
  - $\lozenge = \triangleleft$.
    - It does not always terminate.
  
  - $\lozenge$ = push-pull.
    - It is not correct.
Yet Another Ring Protocol

- Program for agent $i$:

$$\star [(\neg \bigwedge_{j=1}^{n} F_i J) \lor \neg K_i F_{i \oplus 1} l \oplus 1 \rightarrow (i, i \oplus 1)].$$

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

- Properties of this protocol:

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

- Program for agent $i$:

$$
*[(\neg \bigwedge_{j=1}^n F_i J) \lor \neg K_i F_{i \oplus 1} I \oplus 1 \rightarrow (i, i \oplus 1)].
$$

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

- Properties of this protocol:

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

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

- Program for agent $i$:

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

- 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.

- Properties of this protocol:

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>FT</th>
<th>T for $\triangleright$</th>
<th>FT for $\triangleright$</th>
<th>T for $\triangleleft$</th>
<th>FT for $\triangleleft$</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>
A Terminating Ring Protocol, ctd

Program for agent $i$:

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

Non-termination for $\triangleleft$.

$i \triangleleft i \oplus 1, i \triangleleft i \oplus 1, \ldots$

Termination for push-pull.

After each call $(i, i \oplus 1)$ the set

$$\text{Inf} := \{(i, j) \mid \neg K_i F_{i \oplus 1} J\}.$$ decreases.
Protocols for Complete Graphs

Learn New Secrets Protocol ([ADGH14])

- Program for agent $i$:
  $$*\left[\left[ j \in A \neg F_i J \rightarrow (i, j) \right]\right].$$

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

- Properties of this protocol:

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>FT</th>
<th>T for $\triangleright$</th>
<th>FT for $\triangleright$</th>
<th>T for $\triangleleft$</th>
<th>FT for $\triangleleft$</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
Learn New Secrets Protocol for push-pull

- Program for agent $i$:

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

- It can generate the shortest sequences of $2n - 4$ calls.
- Indeed, let $A = \{a, b, c, d, i_1, \ldots, i_{n-4}\}$. Then

\[
\begin{align*}
(a, i_1), (a, i_2), \ldots, (a, i_{n-4}), \\
(a, b), (c, d), (a, d), (b, c), \\
(i_1, b), (i_2, b), \ldots, (i_{n-4}, b)
\end{align*}
\]

  corresponds to a terminating computation.

- It can also generate the longest sequences of $\frac{n(n-1)}{2}$ calls.
- Indeed, take

\[ [2], [3], [4], \ldots, [n], \]

where $[k]$ stands for $(1, k), (2, k), \ldots, (k - 1, k)$. 

Krzysztof R. Apt

Epistemic Gossip Protocols
Hear My Secret Protocol

- Program for agent $i$:

$$*[[\neg K_i F_j I \rightarrow (i, j)]]\text{.}$$

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

- Properties of this protocol:

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

- It can generate the shortest and the longest sequences of calls. Argument the same as for the LNS protocol.
Future Work

- Analyze the protocols for other types of calls, so
  - $ab^-$: every agent $c \neq a, b$ noted that $a$ called $b$,
  - $ab^0$: every agent $c \neq a, b$ noted that some call took place, though not between whom.
  - $ab^+$: every agent $c \neq a, b$ noted that possibly some call took place, though not between whom.

- Introduce a logic in which the correctness proofs can be formally written out.

- Prove (im)possibility results for various protocols concerning generation of (not only) the shortest sequences and for various modes of communication.

- Consider initial situations in which the agents have some partial knowledge of the secrets.
Thank you