Building and verifying a quasi-certification entity over Distributed Hash Tables

Join work with X. Bonnaire, R. Cortes & O. Marin
UPMC (France), UFSM (Chile), NYUS (China)

Fabrice.Kordon@lip6.fr

Certification, why?

Motivations

- Digital filling for tax purpose
  - **Certify that somebody did it before a given deadline**
- Certified emails
  - **Use emails for legal purposes**
- Online game refereeing
- e-voting, etc.

Certification, why?

Motivations

- Digital filling for tax purpose
  - **Certify that somebody did it before a given deadline**
- Certified emails
  - **Use emails for legal purposes**
- Online game refereeing
- e-voting, etc.

Existing Solutions

- Centralized
  - **Public Key Infrastructures (traditional PKI)**
  - **Scaling problem/prone to faults/implementation (atomic multicast)**
- Decentralized
  - **Certification on top of Distributed Hash Tables (DHT)**
  - **Rapidly brings Byzantine consensus (for a 100.0% guarantee)**

Certification, why?

Motivations

- Digital filling for tax purpose
  - **Certify that somebody did it before a given deadline**
- Certified emails
  - **Use emails for legal purposes**
- Online game refereeing
- e-voting, etc.

Objective

(quasi)-certify that a given action has been performed at a certain time

Distributed context (DHT)

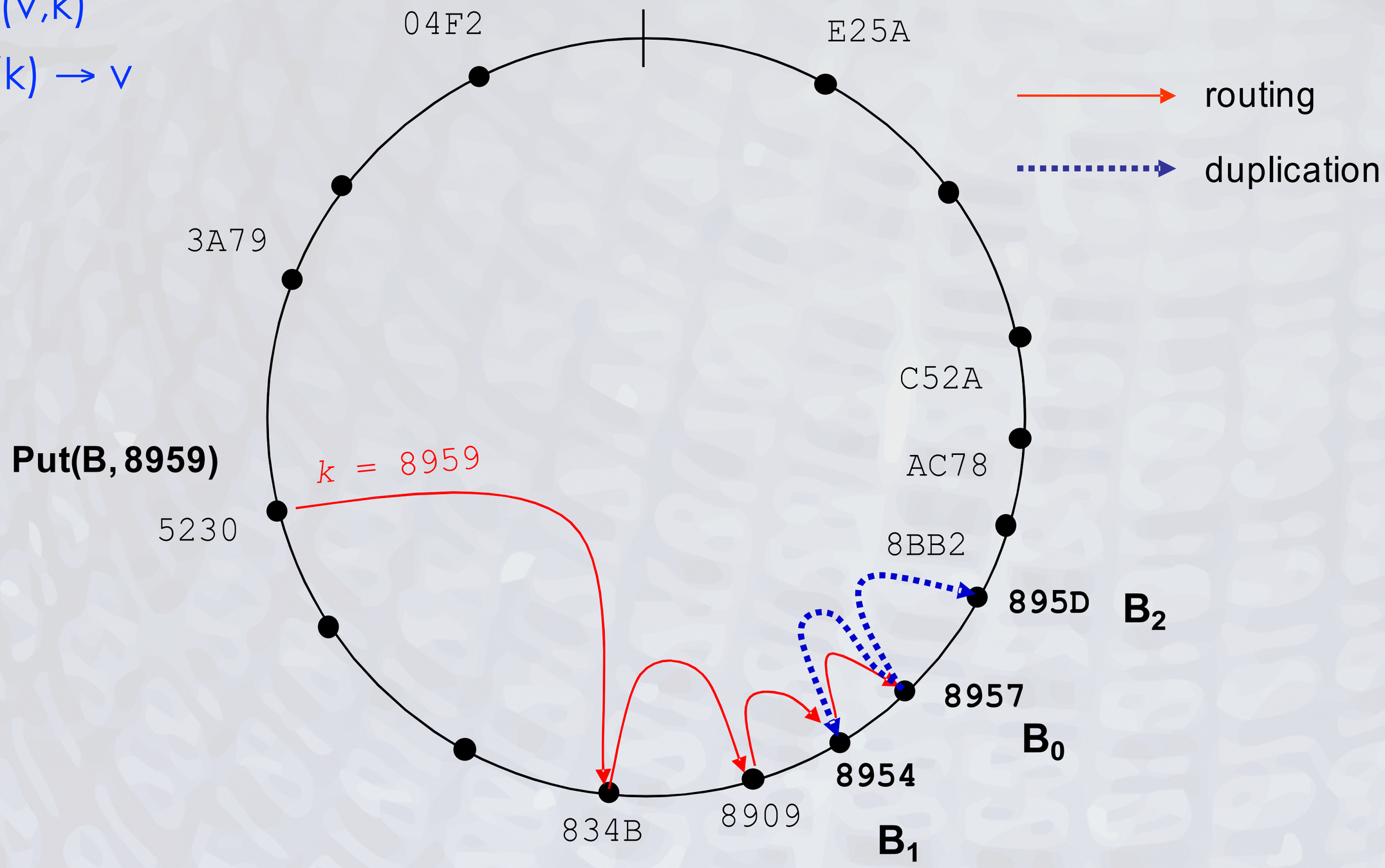
Existing Solutions

- Centralized
  - **Public Key Infrastructures (traditional PKI)**
  - **Scaling problem/prone to faults/implementation (atomic multicast)**
- Decentralized
  - **Certification on top of Distributed Hash Tables (DHT)**
  - **Rapidly brings Byzantine consensus (for a 100.0% guarantee)**

DHT in a nutshell

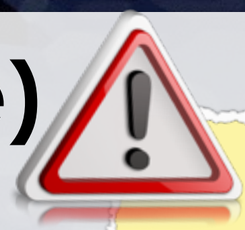
Retrieve data (key + value)

- put (v,k)
- get(k) → v



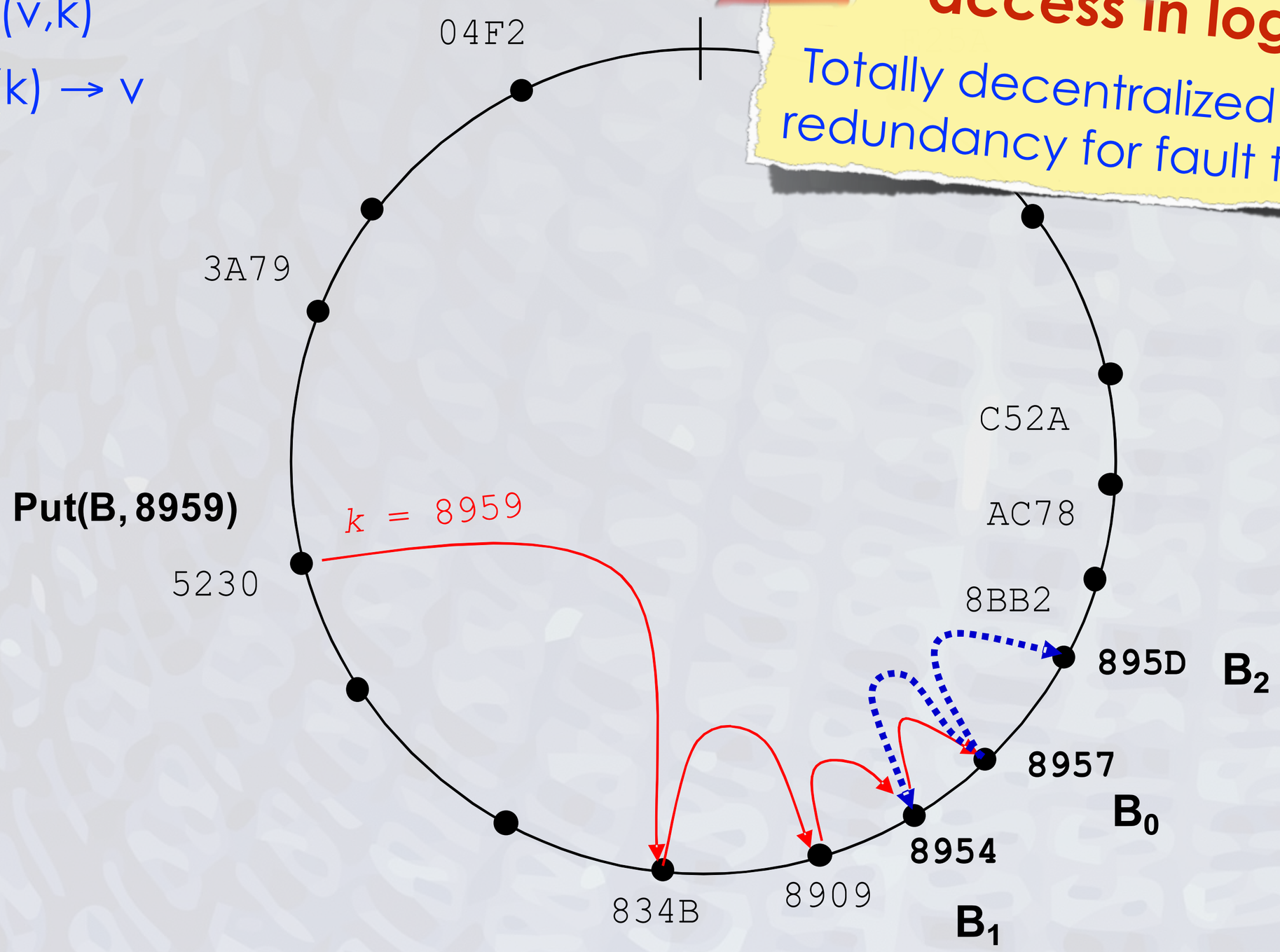
DHT in a nutshell

- **Retrieve data (key + value)**
  - put (v,k)
  - get(k) → v



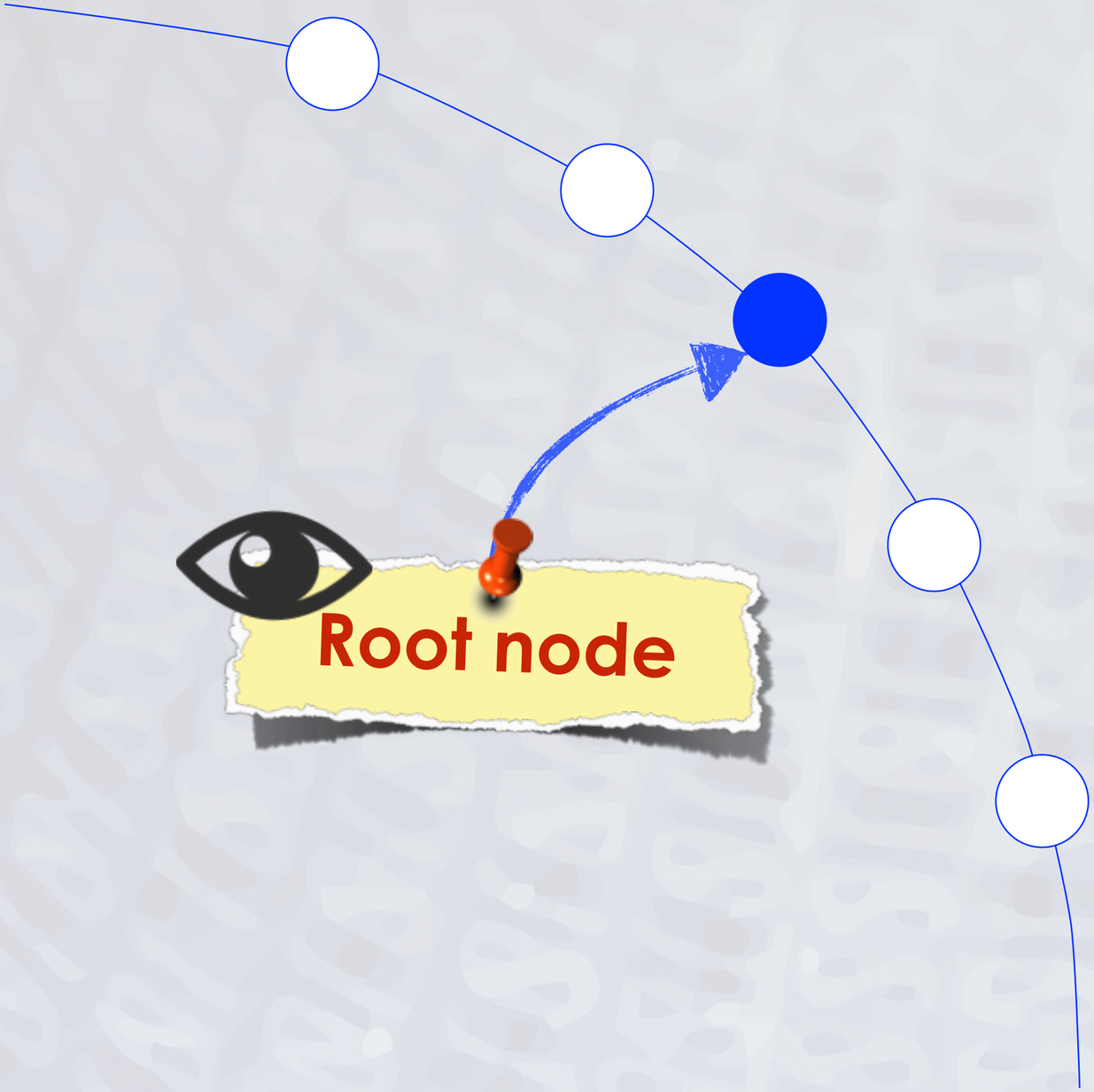
access in log(n)

Totally decentralized + built)in redundancy for fault tolerance



DHT in a nutshell

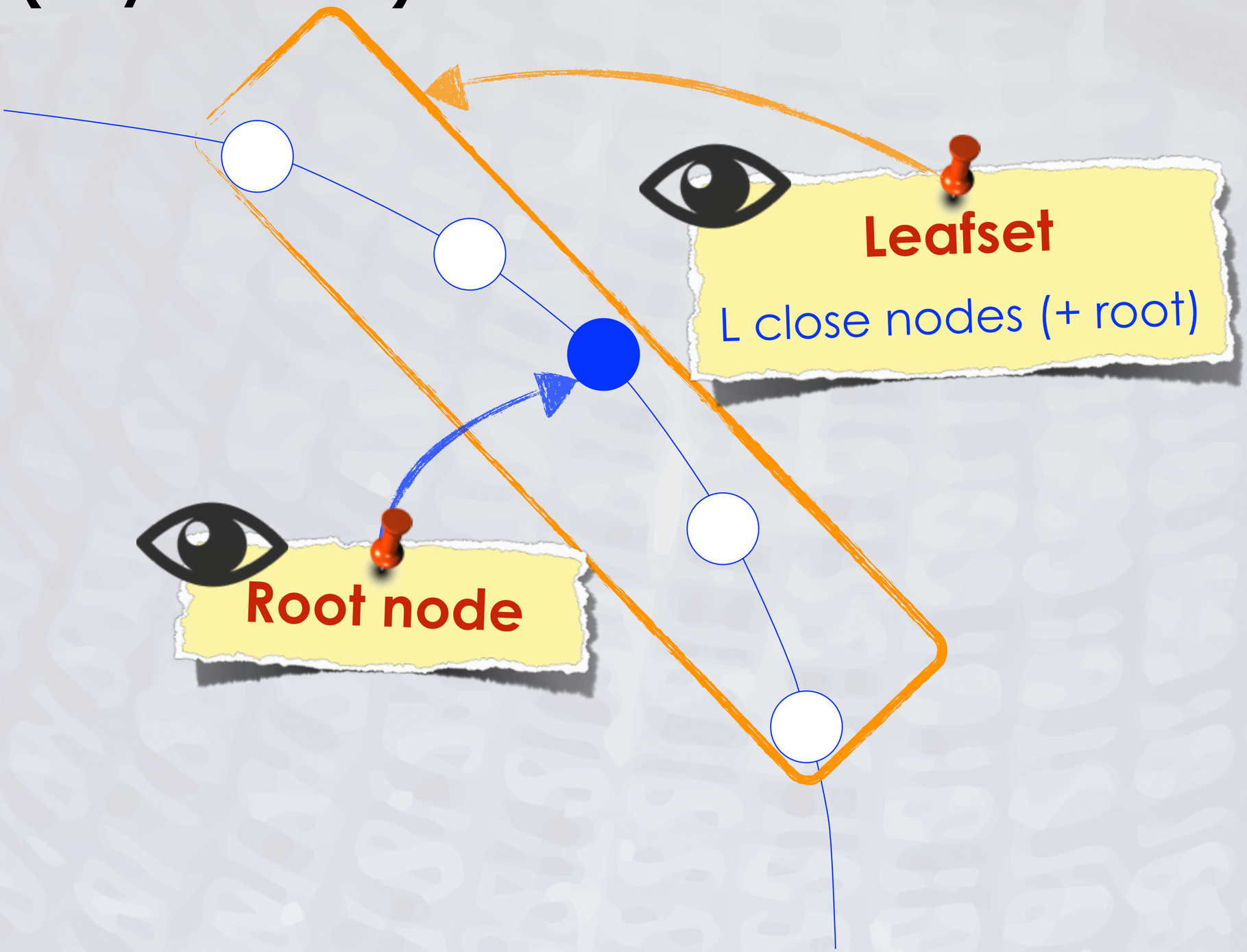
- **Retrieve data (key + value)**
  - put (v,k)
  - get(k) → v



DHT in a nutshell

Retrieve data (key + value)

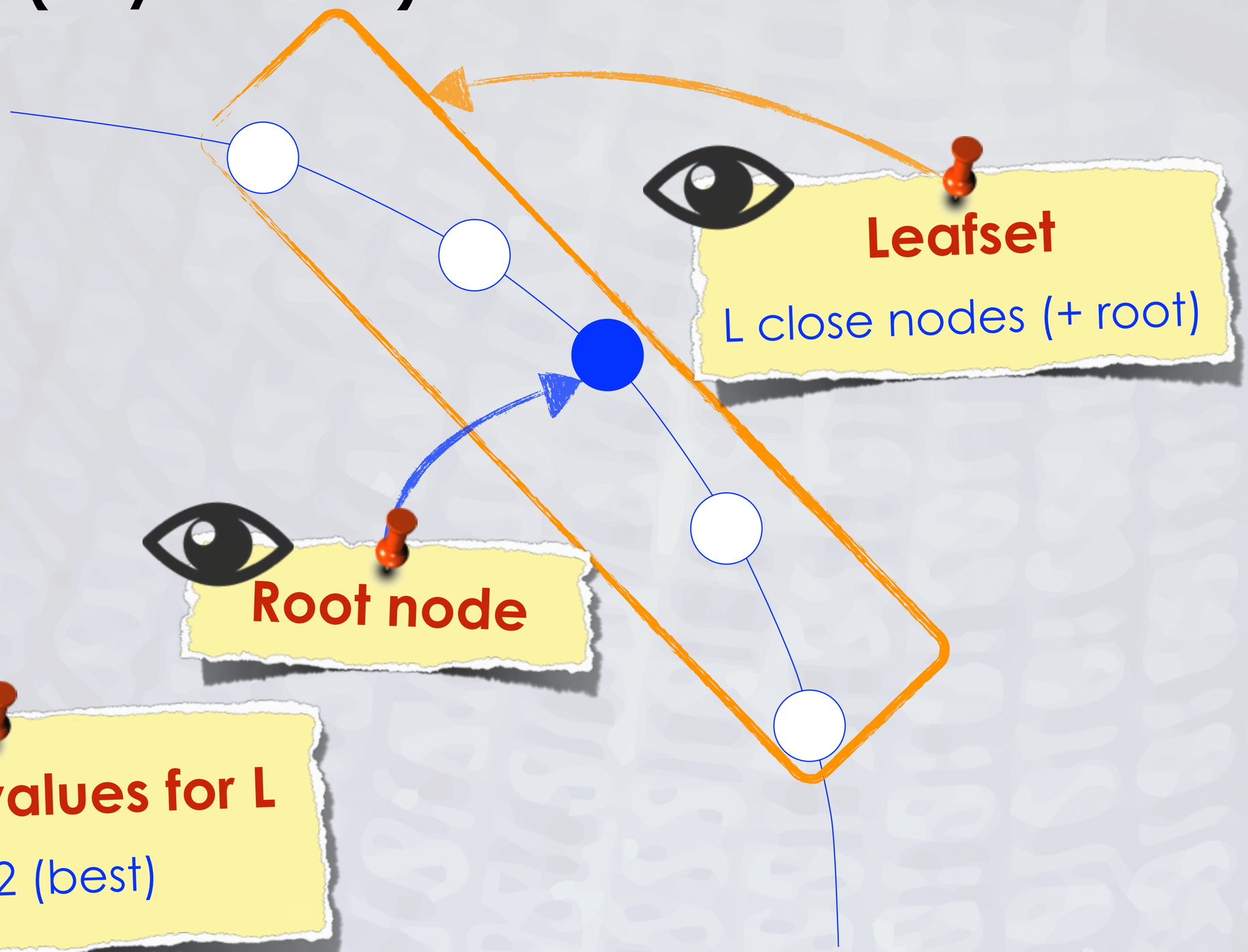
- put (v,k)
- get(k) → v



DHT in a nutshell

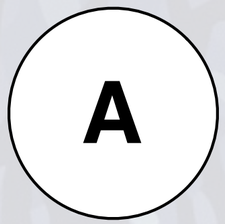
Retrieve data (key + value)

- put (v,k)
- get(k) → v



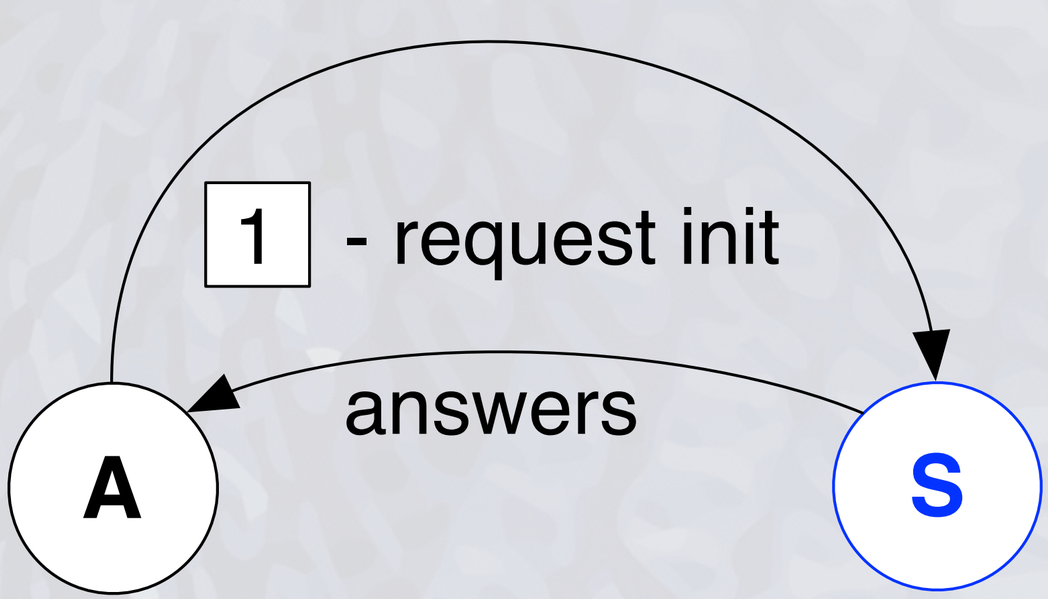
Quasi-certification — entities

- A → an actor performing a service



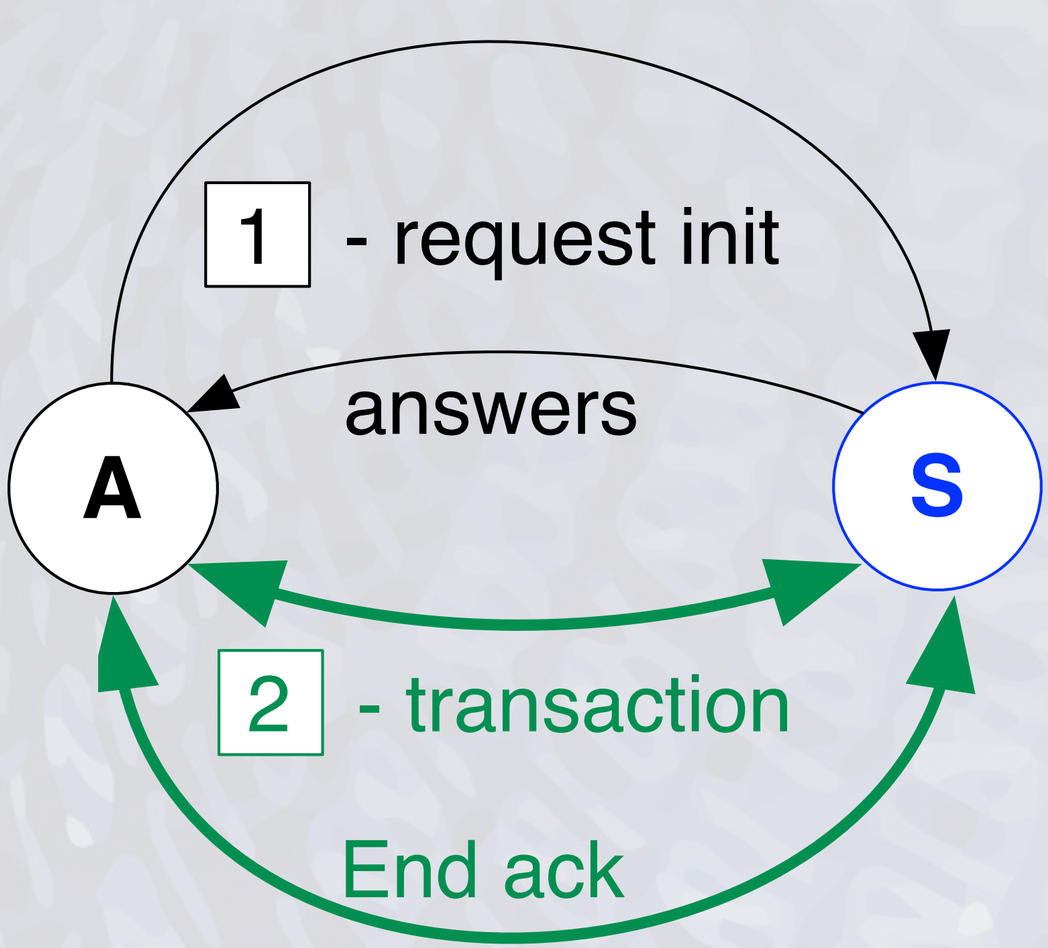
Quasi-certification — entities

- A → an actor performing a service
- S → leafset hash(service) offering the service



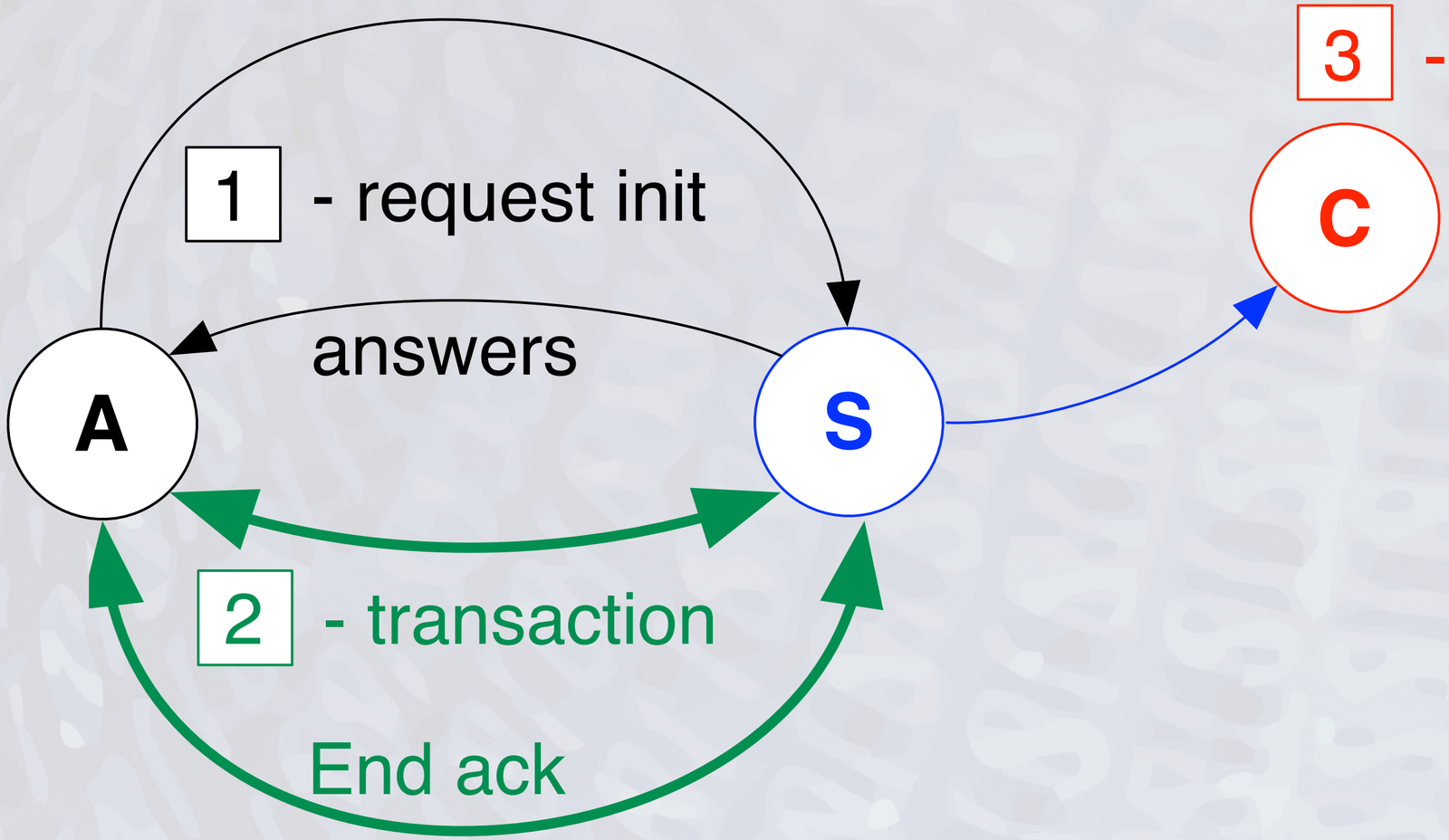
Quasi-certification — entities

- A → an actor performing a service
- S → leafset hash(service) offering the service



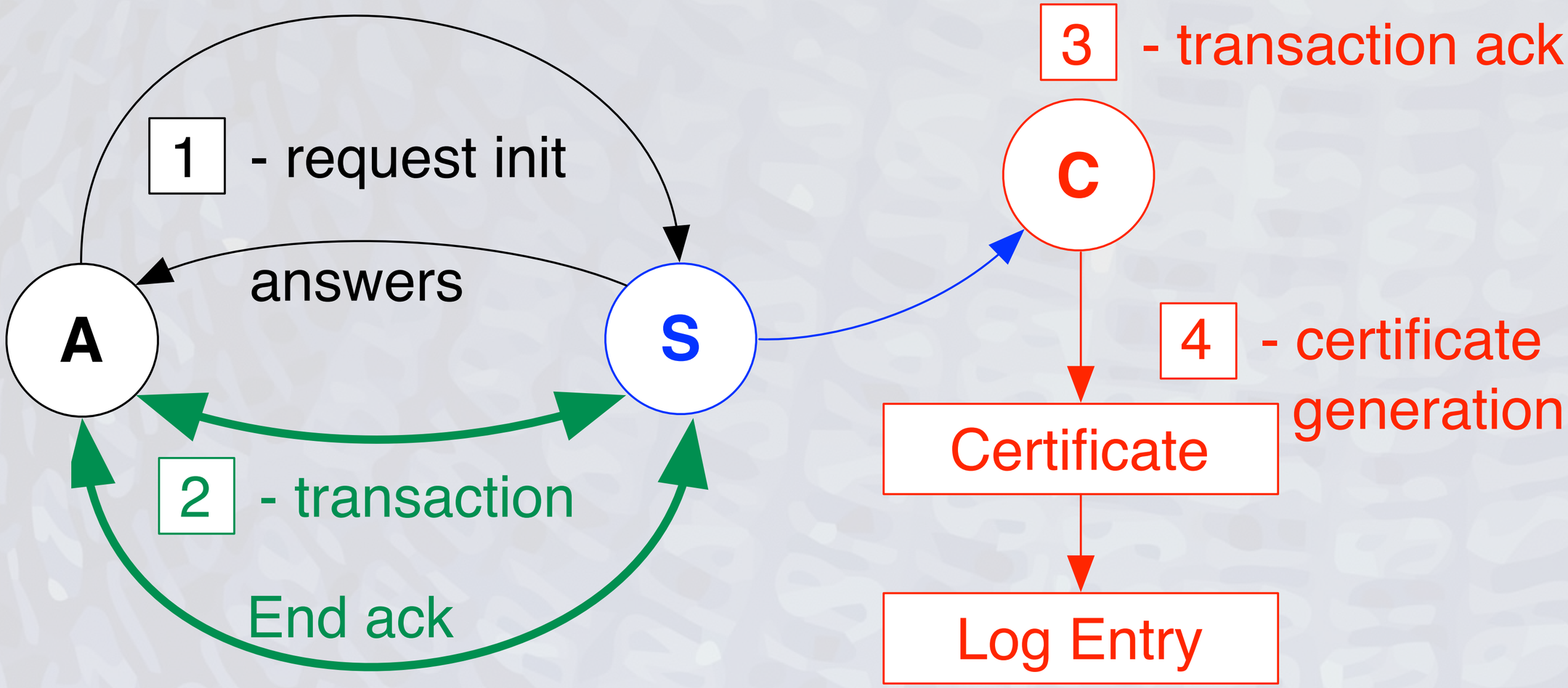
Quasi-certification — entities

- A → an actor performing a service
- S → leafset hash(service) offering the service
- C → certification authority leafset hash(A/service)



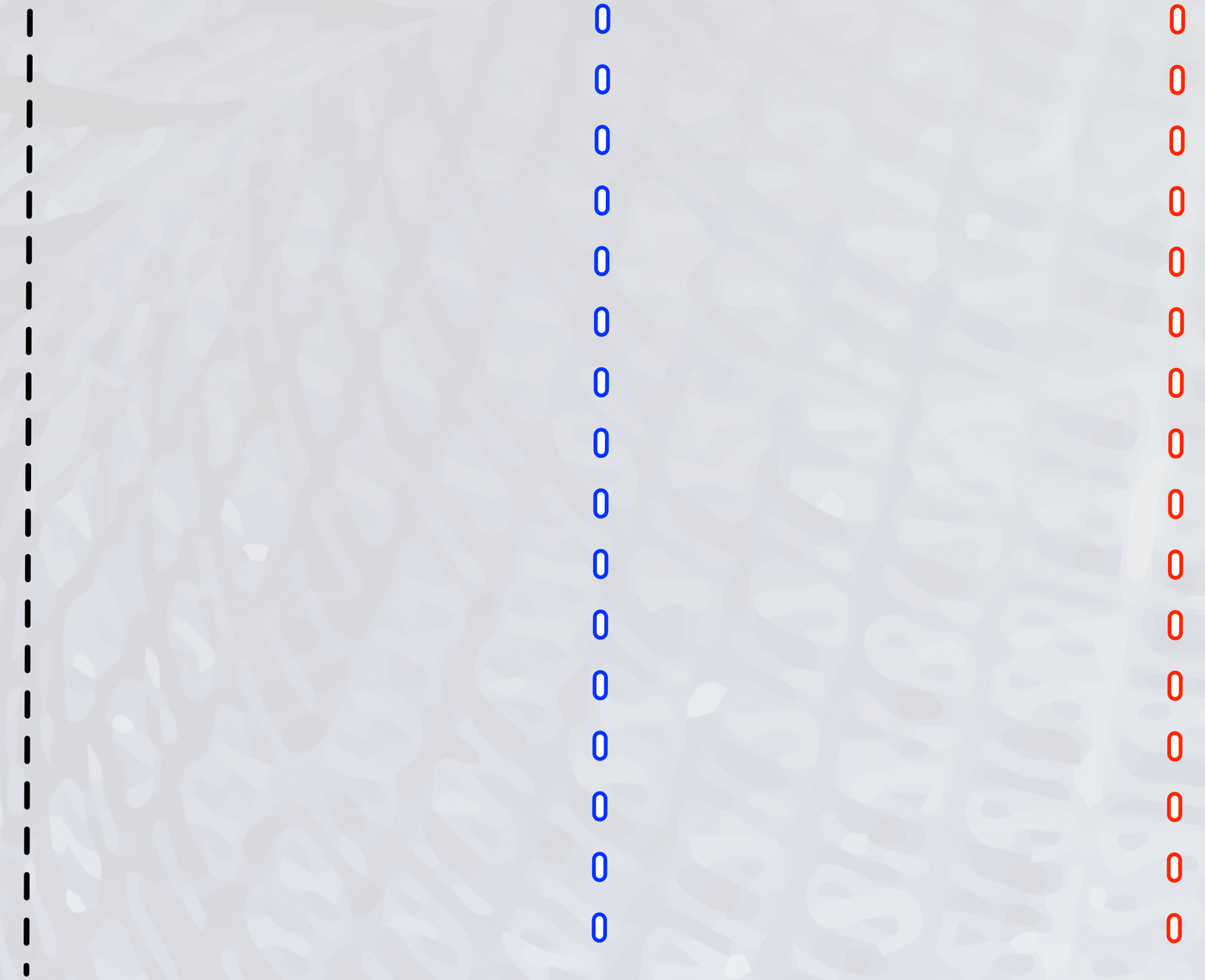
Quasi-certification — entities

- A → an actor performing a service
- S → leafset hash(service) offering the service
- C → certification authority leafset hash(A/service)



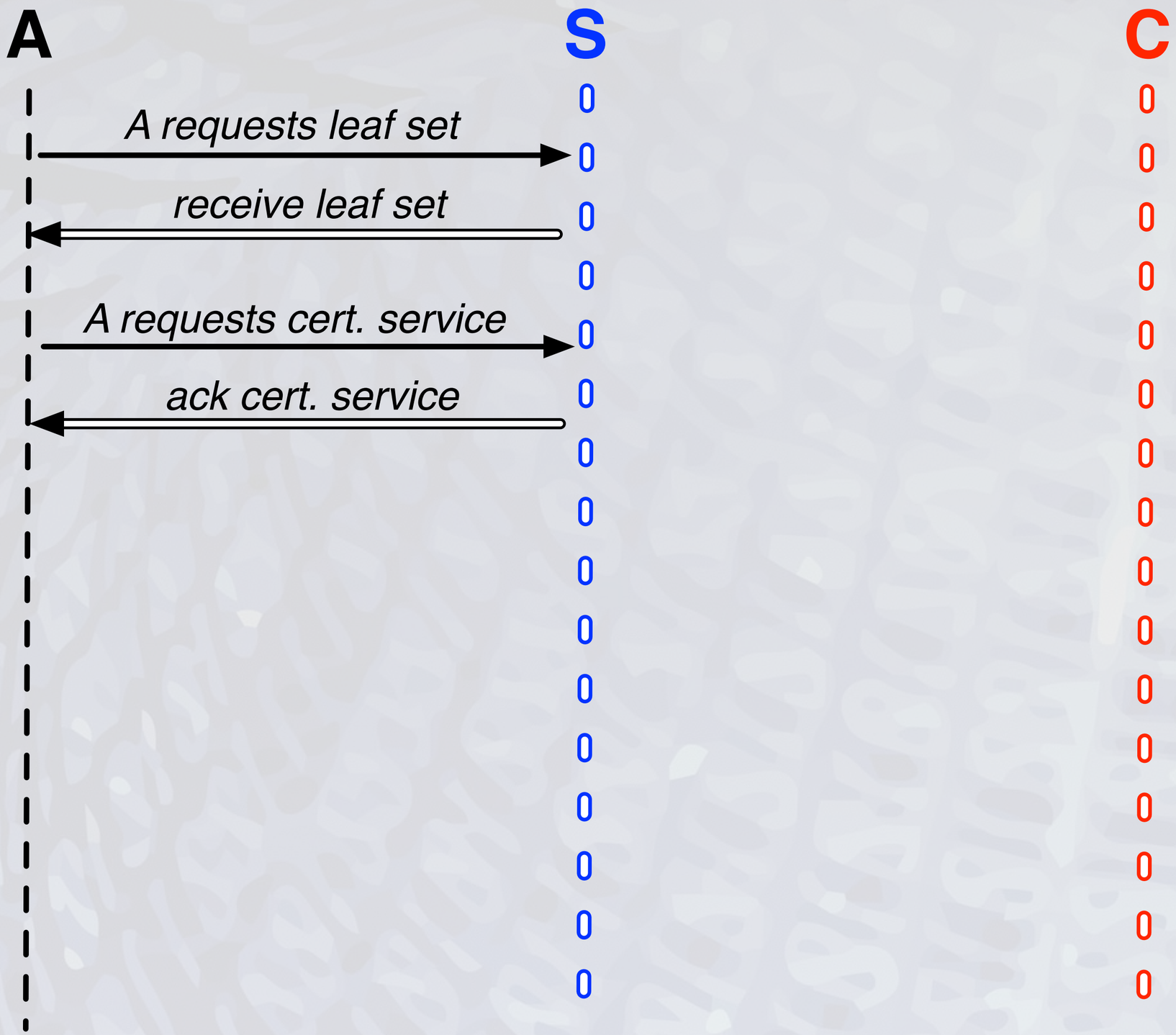
Quasi-certification — protocol structure

5



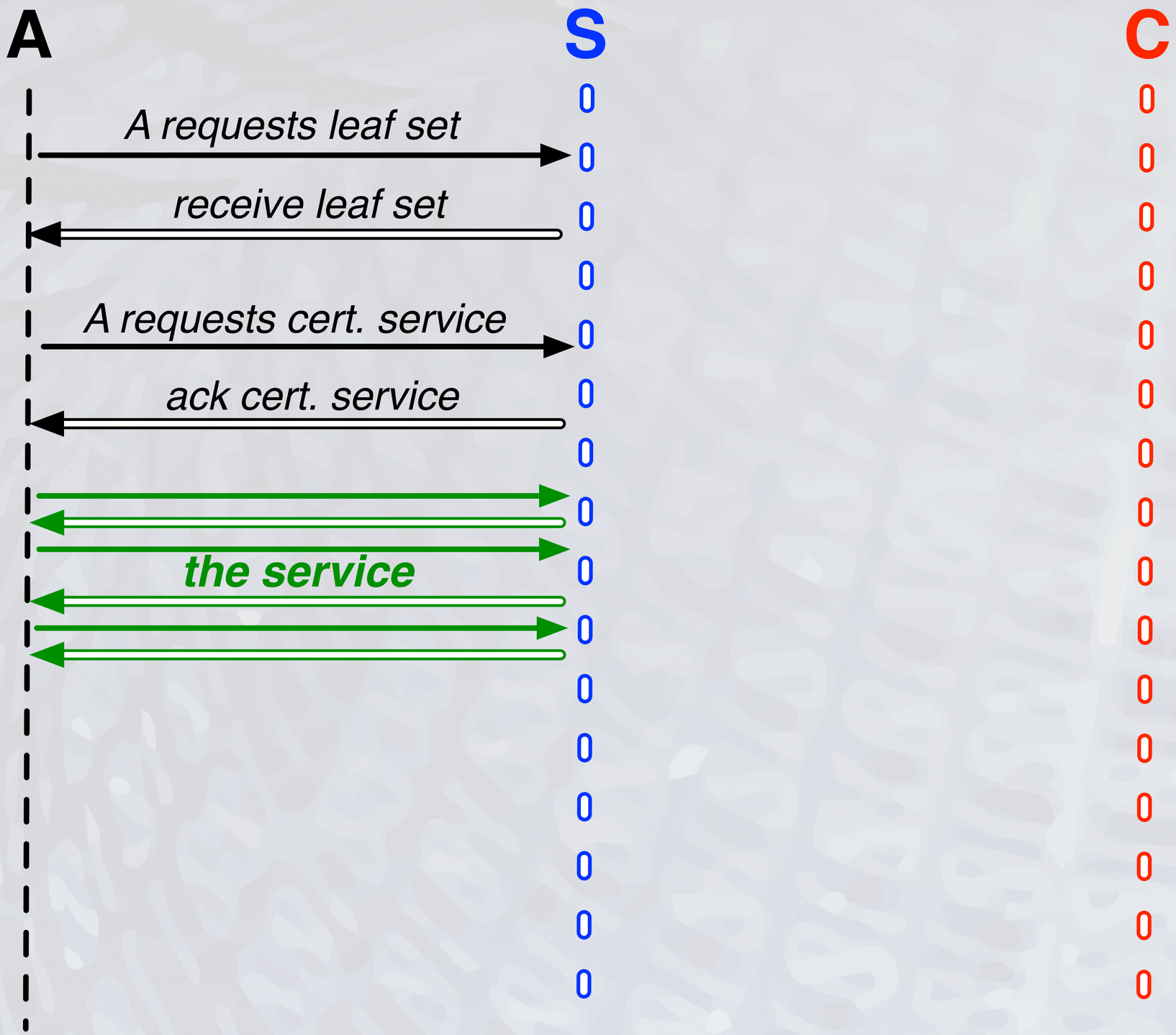
Quasi-certification — protocol structure

5



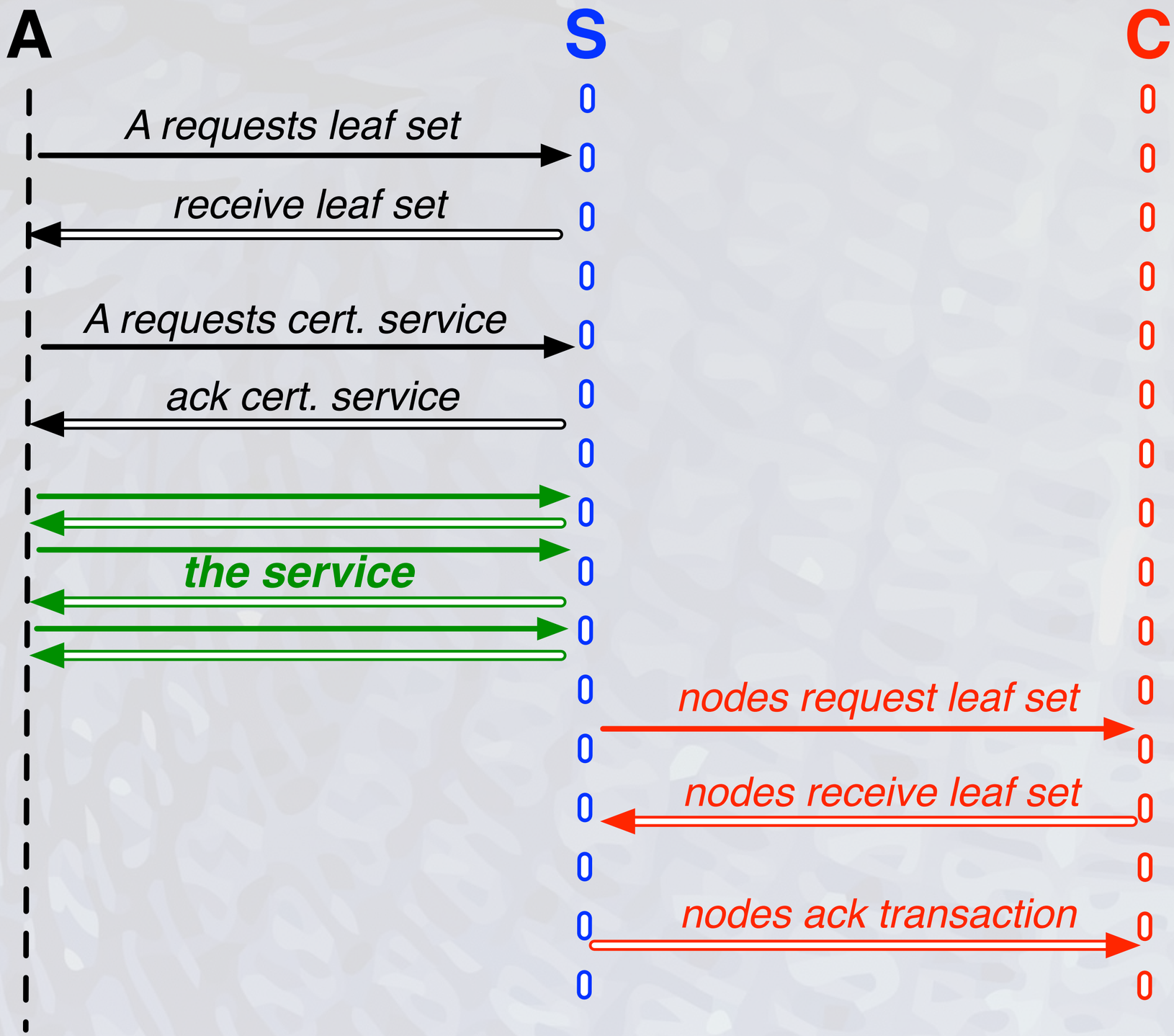
Quasi-certification — protocol structure

5



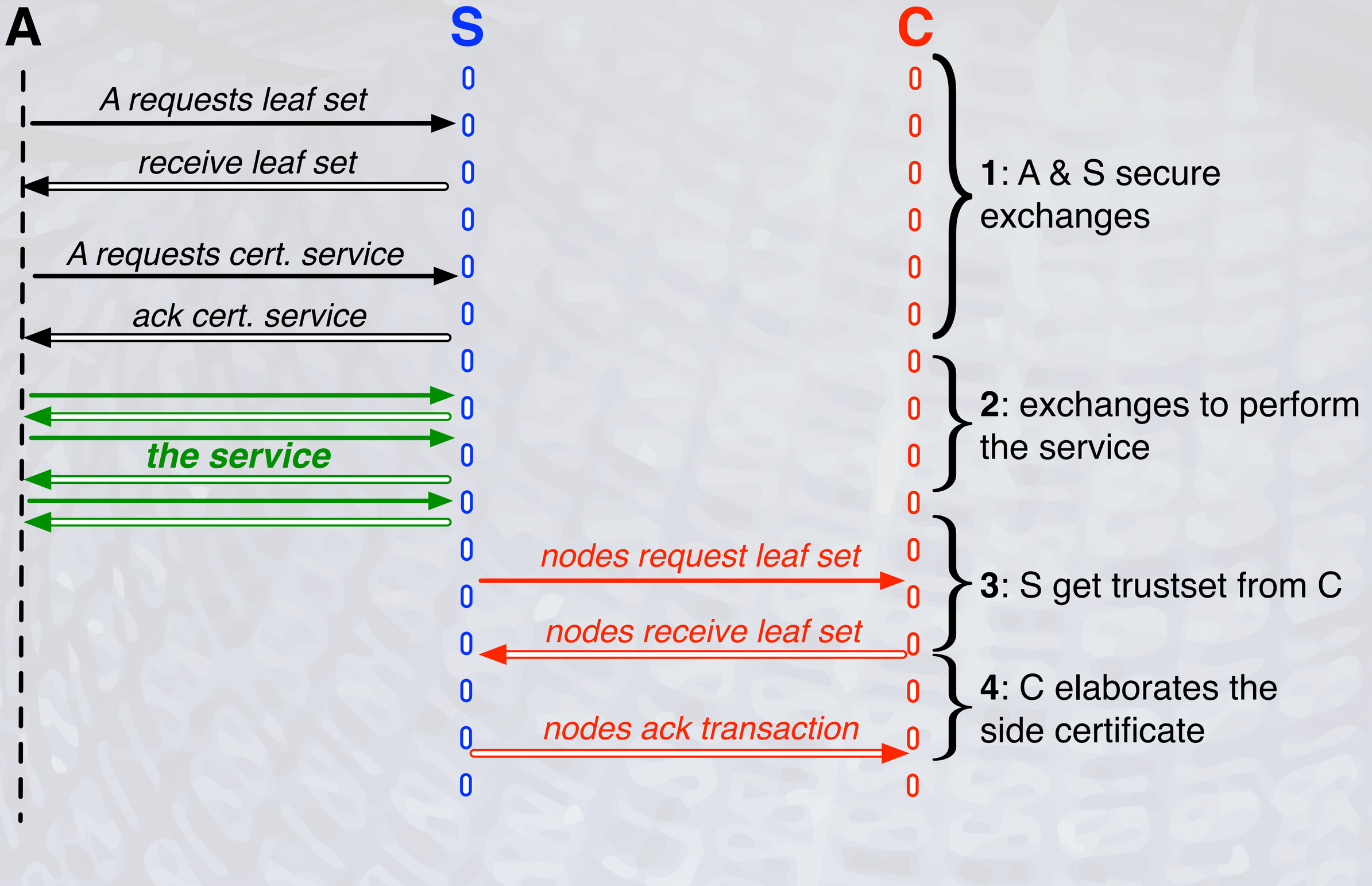
Quasi-certification — protocol structure

5



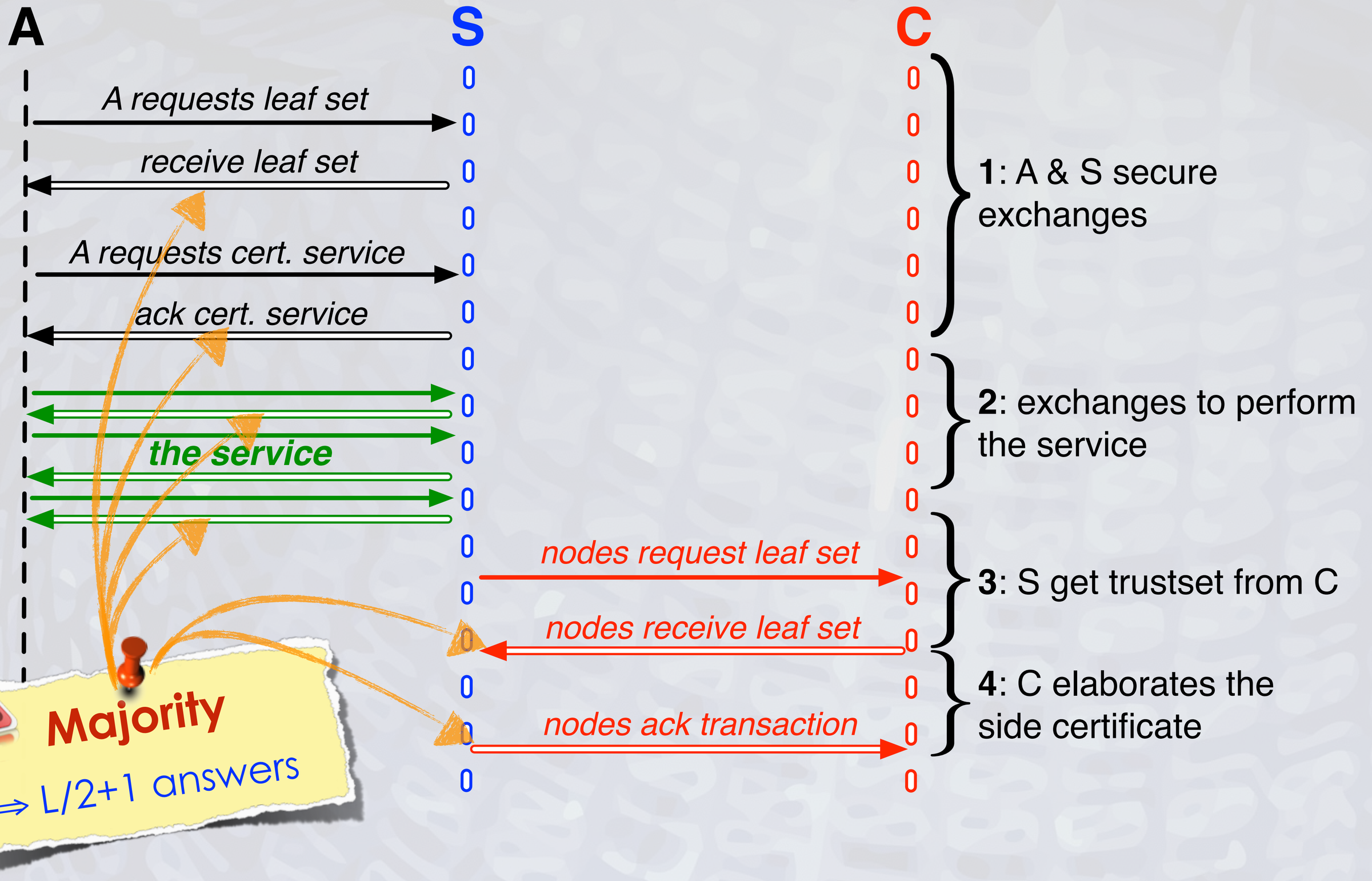
Quasi-certification — protocol structure

5



Quasi-certification — protocol structure

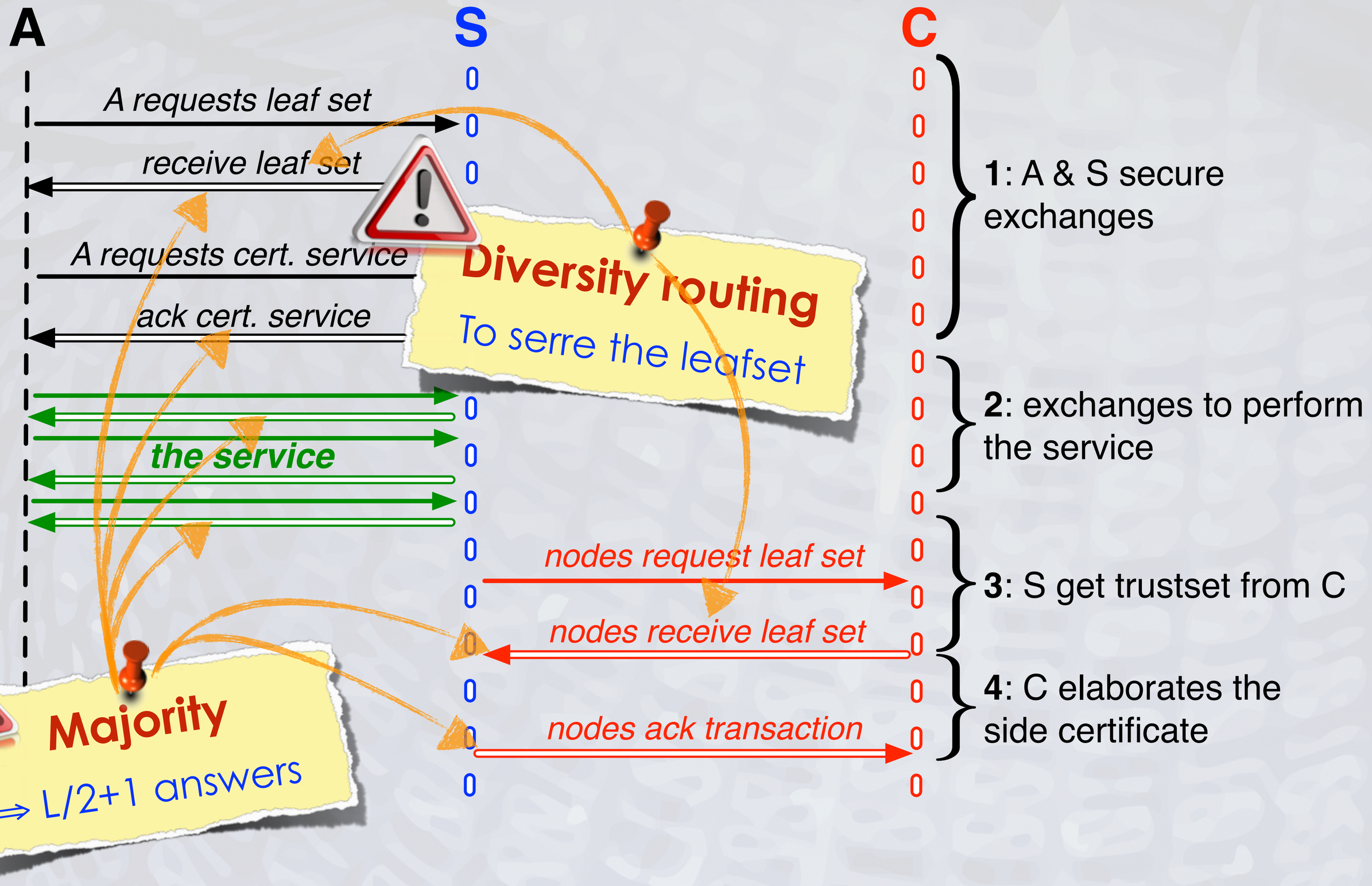
5



F. Kordon - Université P. & M. Curie - CC2016

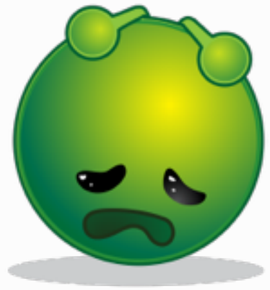
Quasi-certification — protocol structure

5



The verification process

- **Proof (by any method?)**
  - Proven to be undecidable [FLP 85]



Impossibility of Distributed Consensus with One Faulty Process

MICHAEL J. FISCHER

Yale University, New Haven, Connecticut

NANCY A. LYNCH

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

MICHAEL S. PATERSON

University of Warwick, Coventry, England

Abstract. The consensus problem involves an asynchronous system of processes, some of which may be unreliable. The problem is for the reliable processes to agree on a binary value. In this paper, it is shown that every protocol for this problem has the possibility of nontermination, even with only one faulty process. By way of contrast, solutions are known for the synchronous case, the "Byzantine Generals" problem.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols–*protocol architecture*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems–*distributed applications; distributed databases; network operating systems*; C.4 [**Performance of Systems**]: Reliability, Availability, and Serviceability; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation–*parallelism*; H.2.4 [**Database Management**]: Systems–*distributed systems; transaction processing*

General Terms: Algorithms, Reliability, Theory

Additional Key Words and Phrases: Agreement problem, asynchronous system, Byzantine Generals problem, commit problem, consensus problem, distributed computing, fault tolerance, impossibility proof, reliability

1. *Introduction*

The problem of reaching agreement among remote processes is one of the most fundamental problems in distributed computing and is at the core of many

Editing of this paper was performed by guest editor S. L. Graham. The Editor-in-Chief of JACM did not participate in the processing of the paper.

This work was supported in part by the Office of Naval Research under Contract N00014-82-K-0154, by the Office of Army Research under Contract DAAG29-79-C-0155, and by the National Science Foundation under Grants MCS-7924370 and MCS-8116678.

This work was originally presented at the 2nd ACM Symposium on Principles of Database Systems, March 1983.

Authors' present addresses: M. J. Fischer, Department of Computer Science, Yale University, P.O. Box 2158, Yale Station, New Haven, CT 06520; N. A. Lynch, Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139; M. S. Paterson, Department of Computer Science, University of Warwick, Coventry CV4 7AL, England

The verification process

- **Proof (by any method?)**
  - Proven to be undecidable [FLP 85]
- **So what?**
  - Being pragmatic
  - Going for «quasi»

Impossibility of Distributed Consensus with One Faulty Process

MICHAEL J. FISCHER

Yale University, New Haven, Connecticut

NANCY A. LYNCH

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

MICHAEL S. PATERSON

University of Warwick, Coventry, England

Abstract. The consensus problem involves an asynchronous system of processes, some of which may be unreliable. The problem is for the reliable processes to agree on a binary value. In this paper, it is shown that every protocol for this problem has the possibility of nontermination, even with only one faulty process. By way of contrast, solutions are known for the synchronous case, the "Byzantine Generals" problem.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols–*protocol architecture*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems–*distributed applications; distributed databases; network operating systems*; C.4 [**Performance of Systems**]: Reliability, Availability, and Serviceability; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation–*parallelism*; H.2.4 [**Database Management**]: Systems–*distributed systems*; *transaction processing*

General Terms: Algorithms, Reliability, Theory

Additional Key Words and Phrases: Agreement problem, asynchronous system, Byzantine Generals problem, commit problem, consensus problem, distributed computing, fault tolerance, impossibility proof, reliability

1. *Introduction*

The problem of reaching agreement among remote processes is one of the most fundamental problems in distributed computing and is at the core of many

Editing of this paper was performed by guest editor S. L. Graham. The Editor-in-Chief of JACM did not participate in the processing of the paper.

This work was supported in part by the Office of Naval Research under Contract N00014-82-K-0154, by the Office of Army Research under Contract DAAG29-79-C-0155, and by the National Science Foundation under Grants MCS-7924370 and MCS-8116678.

This work was originally presented at the 2nd ACM Symposium on Principles of Database Systems, March 1983.

Authors' present addresses: M. J. Fischer, Department of Computer Science, Yale University, P.O. Box 2158, Yale Station, New Haven, CT 06520; N. A. Lynch, Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139; M. S. Paterson, Department of Computer Science, University of Warwick, Coventry CV4 7AL, England

Journal of the Association for Computing Machinery, Vol. 32, No. 2, April 1985, pp. 374–382.

The verification process

Proof (by any method?)

- Proven to be undecidable [FLP 85]

So what?

- Being pragmatic
- Going for «quasi»

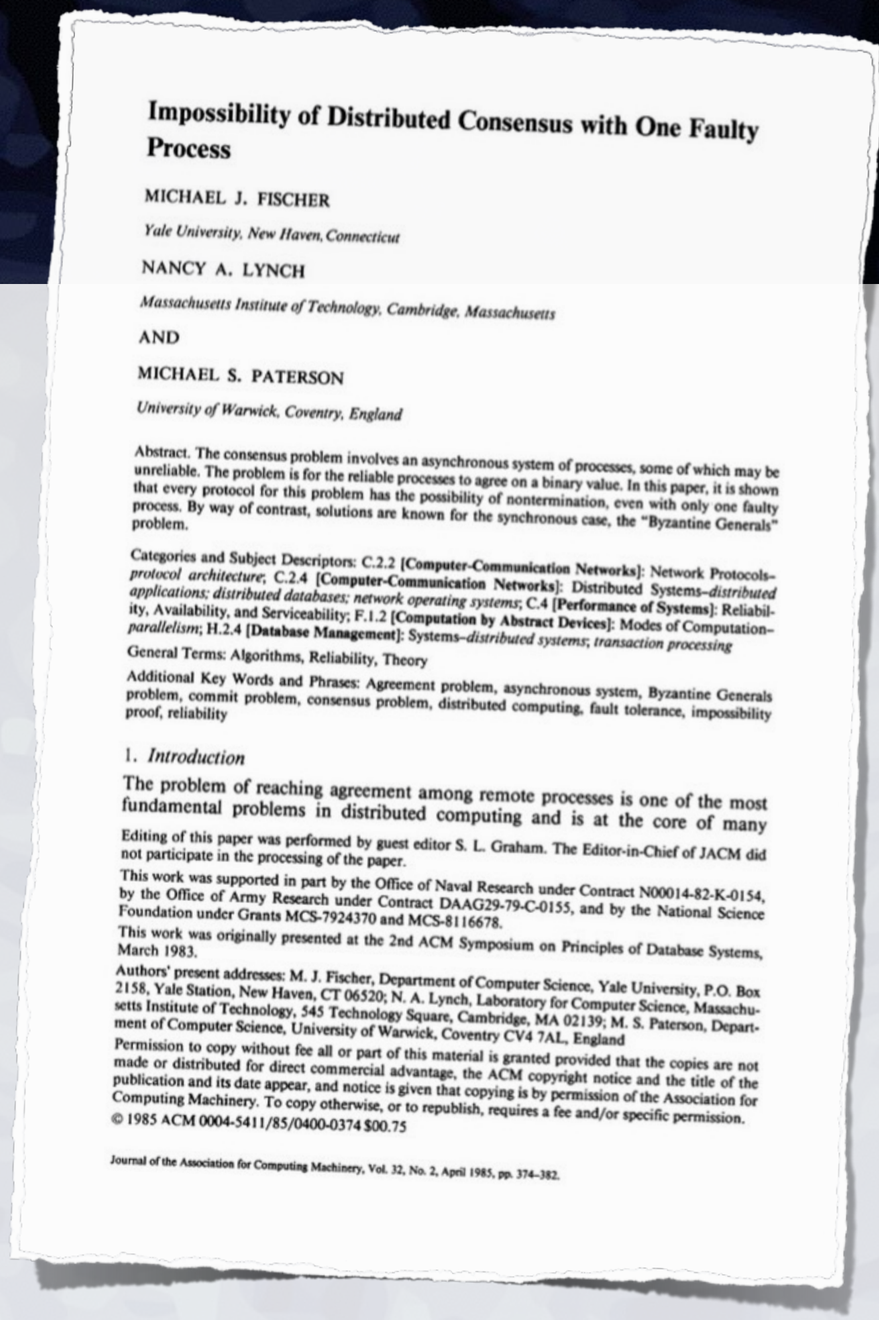
Two steps

1. modeling the protocol in a perfect world (no error)

   Use of Petri nets

2. Probabilistic analysis to evaluate the failure rate

   Use of a classical fault model,
   building a formula + numeric evaluation



Impossibility of Distributed Consensus with One Faulty Process

MICHAEL J. FISCHER

Yale University, New Haven, Connecticut

NANCY A. LYNCH

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

MICHAEL S. PATERSON

University of Warwick, Coventry, England

Abstract. The consensus problem involves an asynchronous system of processes, some of which may be unreliable. The problem is for the reliable processes to agree on a binary value. In this paper, it is shown that every protocol for this problem has the possibility of nontermination, even with only one faulty process. By way of contrast, solutions are known for the synchronous case, the "Byzantine Generals" problem.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols–*protocol architecture*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems–*distributed applications; distributed databases; network operating systems*; C.4 [**Performance of Systems**]: Reliability, Availability, and Serviceability; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation–*parallelism*; H.2.4 [**Database Management**]: Systems–*distributed systems*; *transaction processing*

General Terms: Algorithms, Reliability, Theory

Additional Key Words and Phrases: Agreement problem, asynchronous system, Byzantine Generals problem, commit problem, consensus problem, distributed computing, fault tolerance, impossibility proof, reliability

1. *Introduction*

The problem of reaching agreement among remote processes is one of the most fundamental problems in distributed computing and is at the core of many

Editing of this paper was performed by guest editor S. L. Graham. The Editor-in-Chief of JACM did not participate in the processing of the paper.

This work was supported in part by the Office of Naval Research under Contract N00014-82-K-0154, by the Office of Army Research under Contract DAAG29-79-C-0155, and by the National Science Foundation under Grants MCS-7924370 and MCS-8116678.

This work was originally presented at the 2nd ACM Symposium on Principles of Database Systems, March 1983.

Authors' present addresses: M. J. Fischer, Department of Computer Science, Yale University, P.O. Box 2158, Yale Station, New Haven, CT 06520; N. A. Lynch, Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139; M. S. Paterson, Department of Computer Science, University of Warwick, Coventry CV4 7AL, England

Journal of the Association for Computing Machinery, Vol. 32, No. 2, April 1985, pp. 374–382.

Modeling the protocol (step 1)

Hypotheses

H_1: perfect world

H_2: service reduced to 1 interaction

H_3: L+1 answer requested instead of L/2+1

- **Symmetric net with Bags?**

Modeling the protocol (step 1)

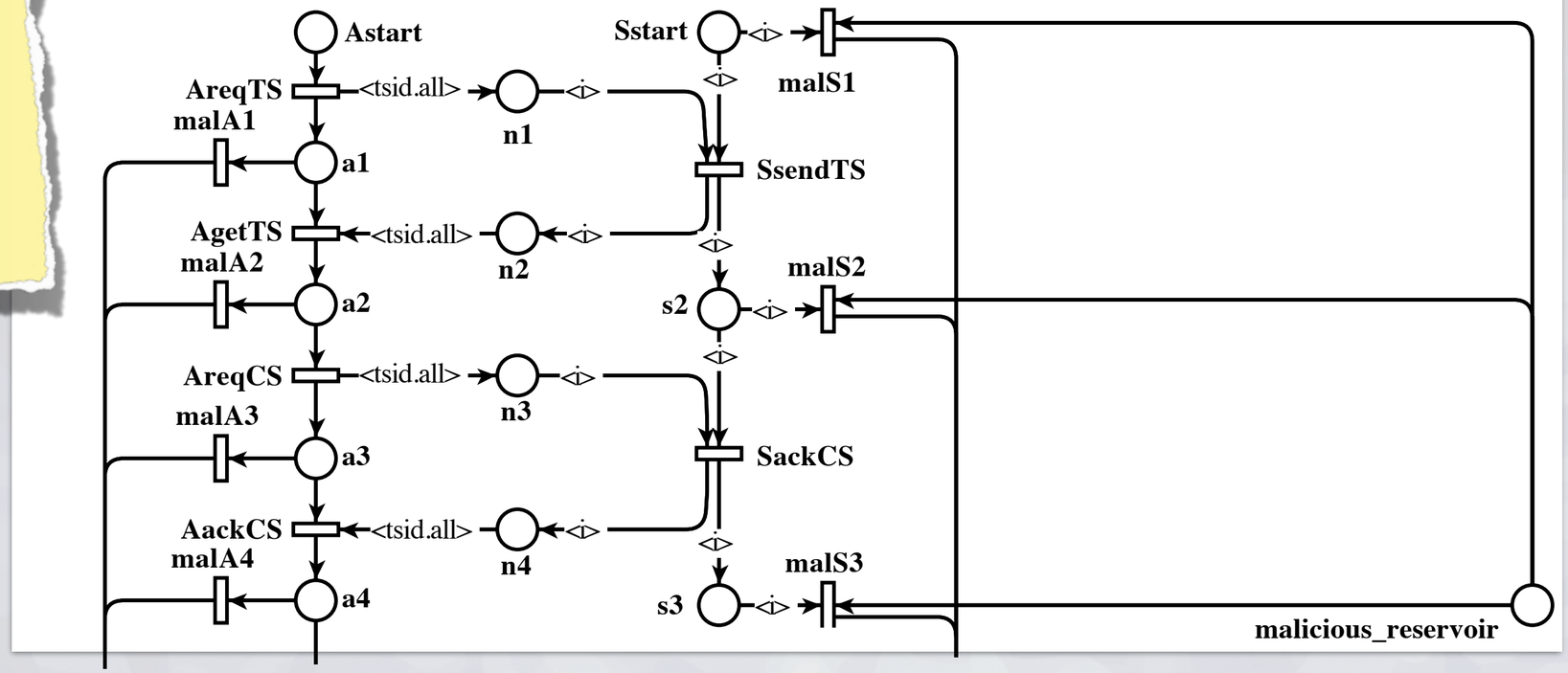
Types and variables

```
type tsid is 0..L;
type tsidxtsid is <tsid, tsid>;

var i in tsid;
```

Modeling the protocol (step 1)

Types and variables

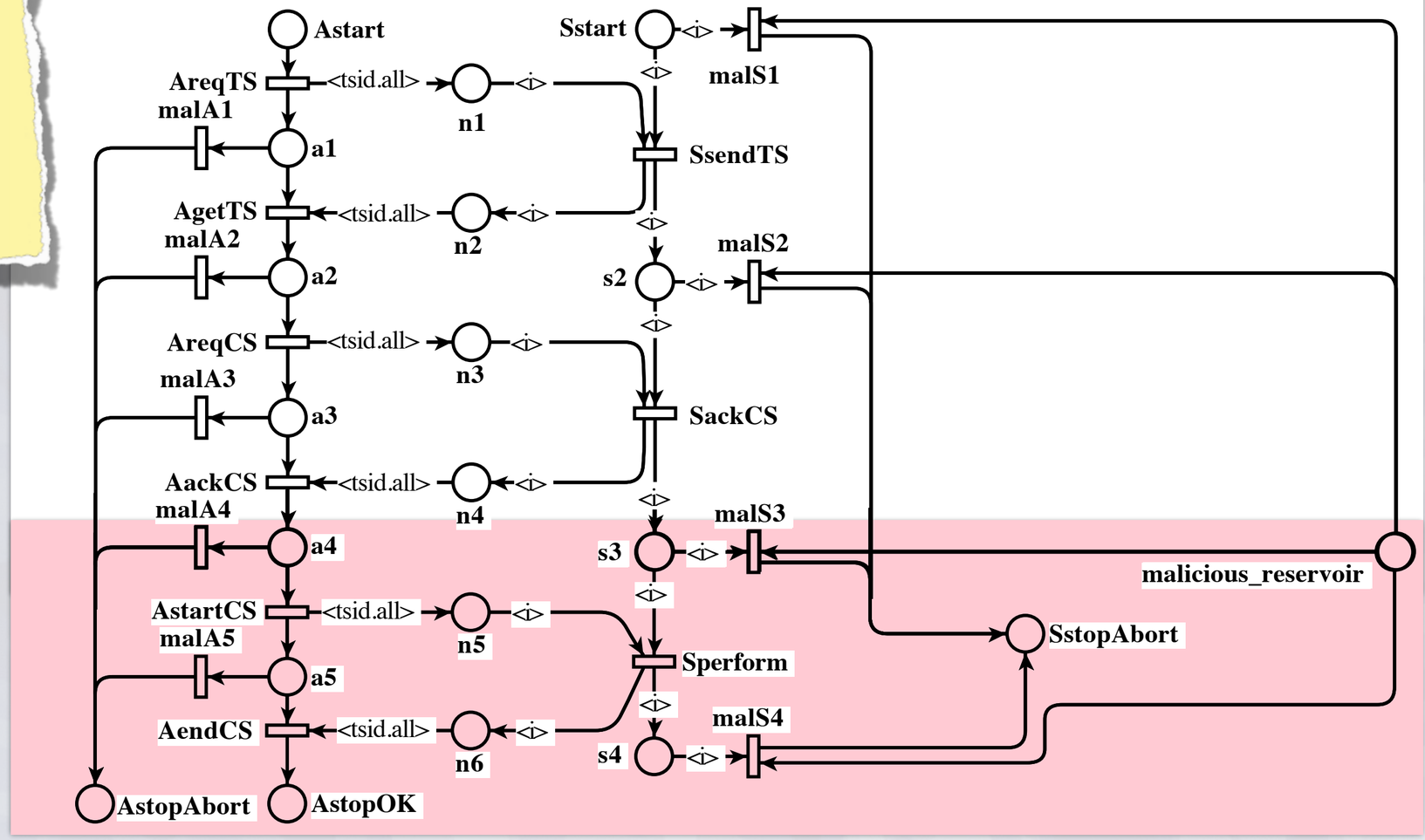
```
type tsid is 0..L;
type tsidxtsid is <tsid, tsid>;

var i in tsid;
```



Modeling the protocol (step 1)

Types and variables

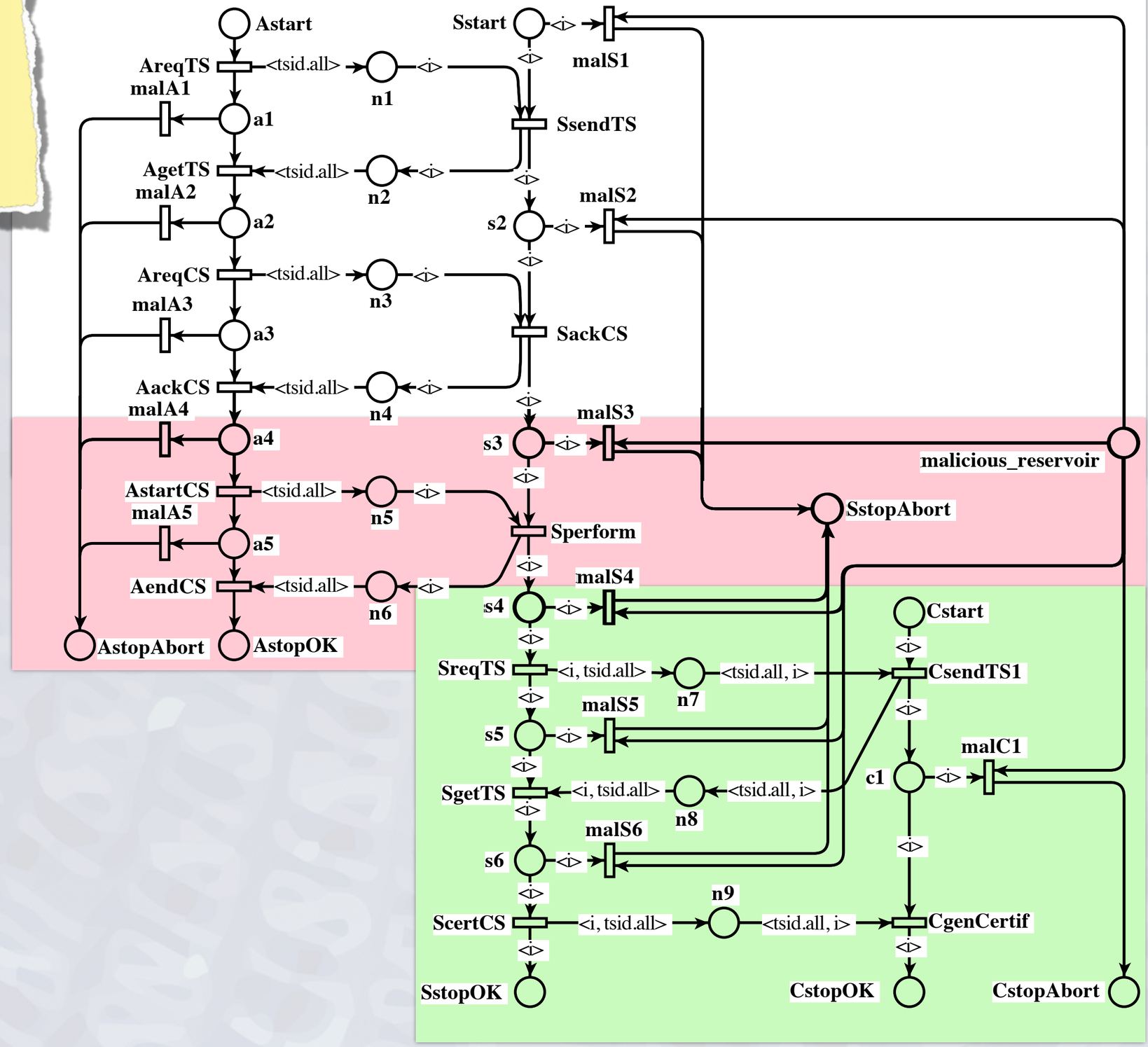
```
type tsid is 0..L;
type tsidxtsid is <tsid, tsid>;

var i in tsid;
```



Modeling the protocol (step 1)

Types and variables

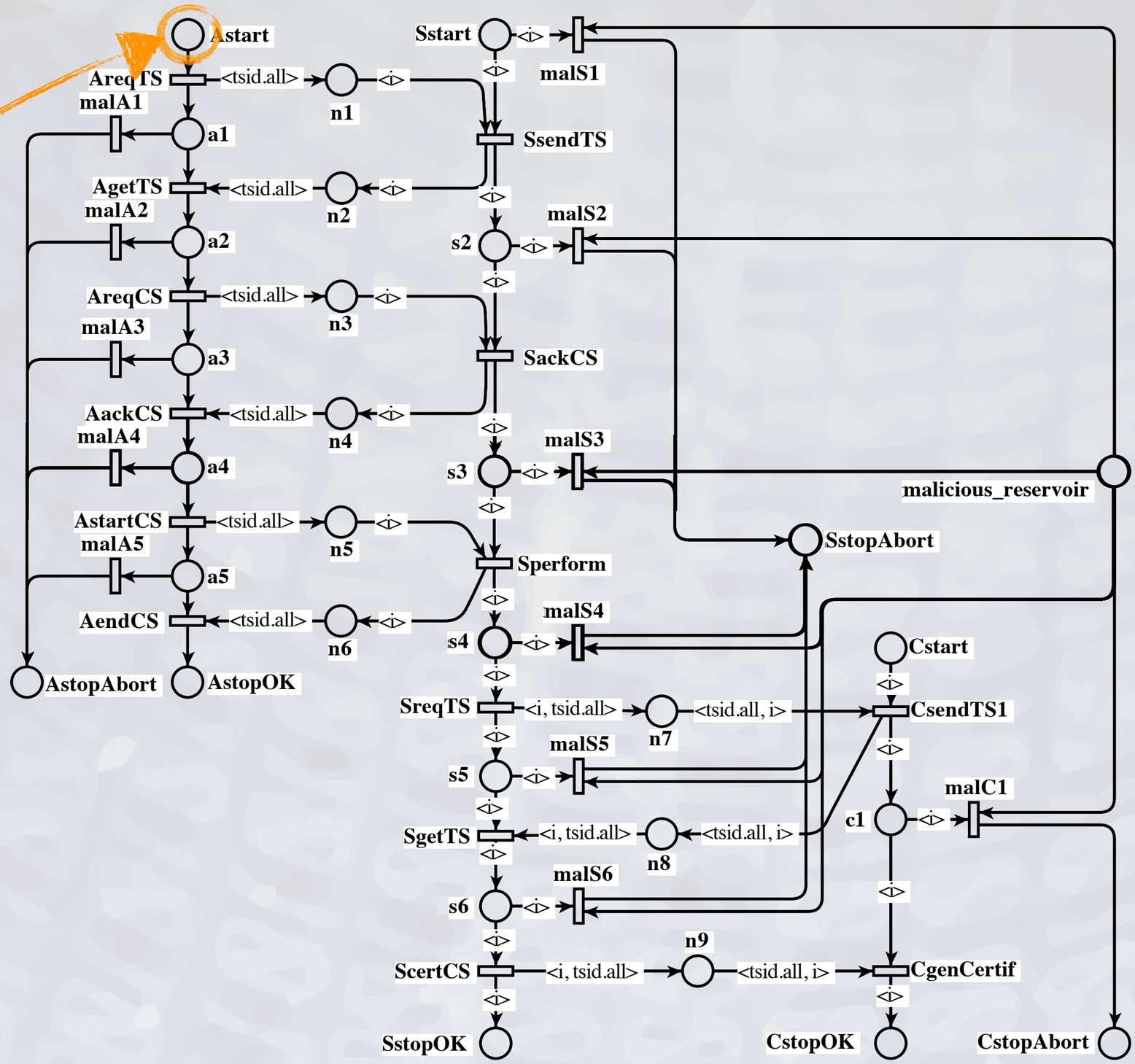
```
type tsid is 0..L;
type tsidxtsid is <tsid, tsid>;

var i in tsid;
```



Modeling the protocol (step 1)

Types and variables

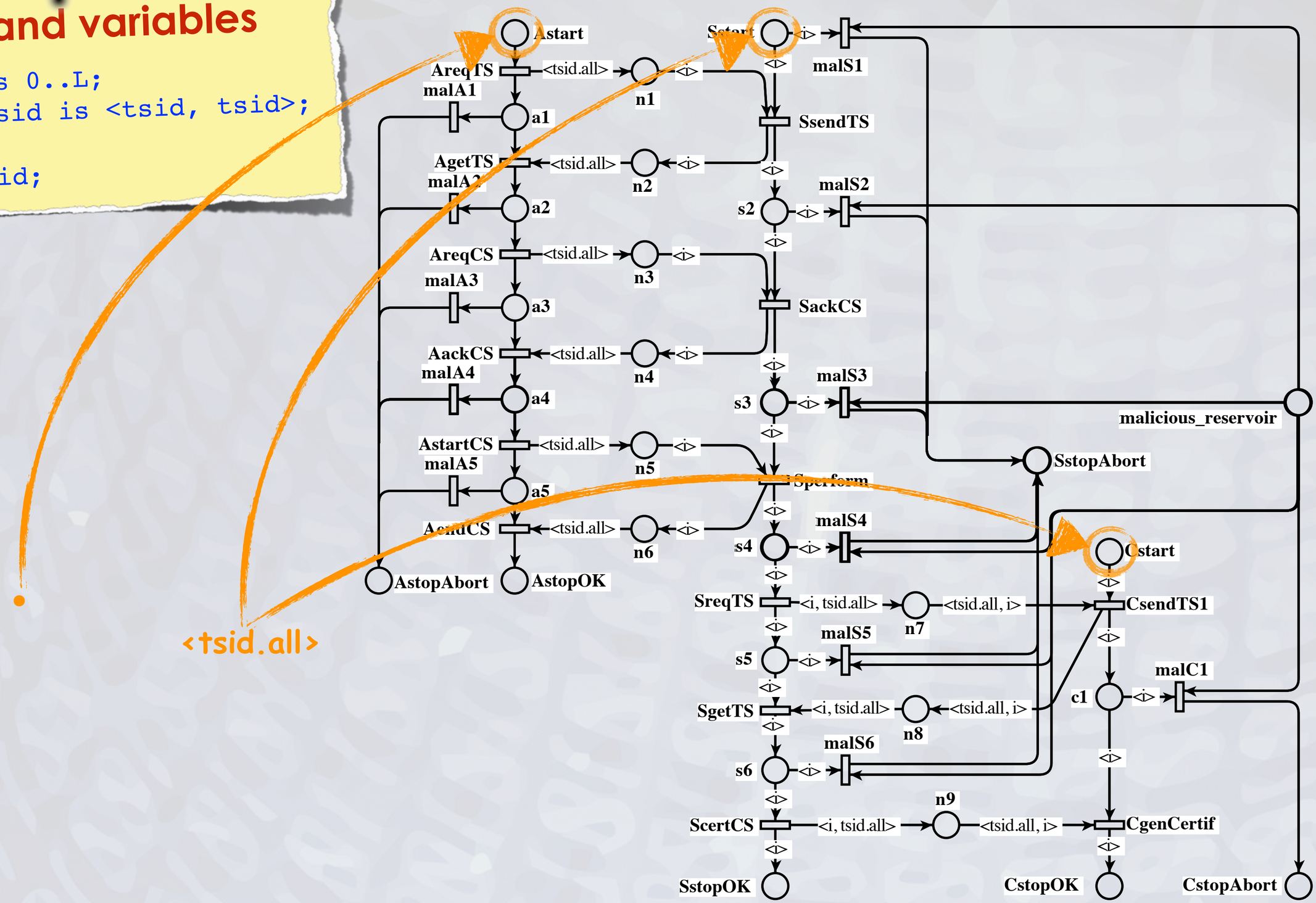
```
type tsid is 0..L;
type tsidxtsid is <tsid, tsid>;

var i in tsid;
```



Modeling the protocol (step 1)

Types and variables

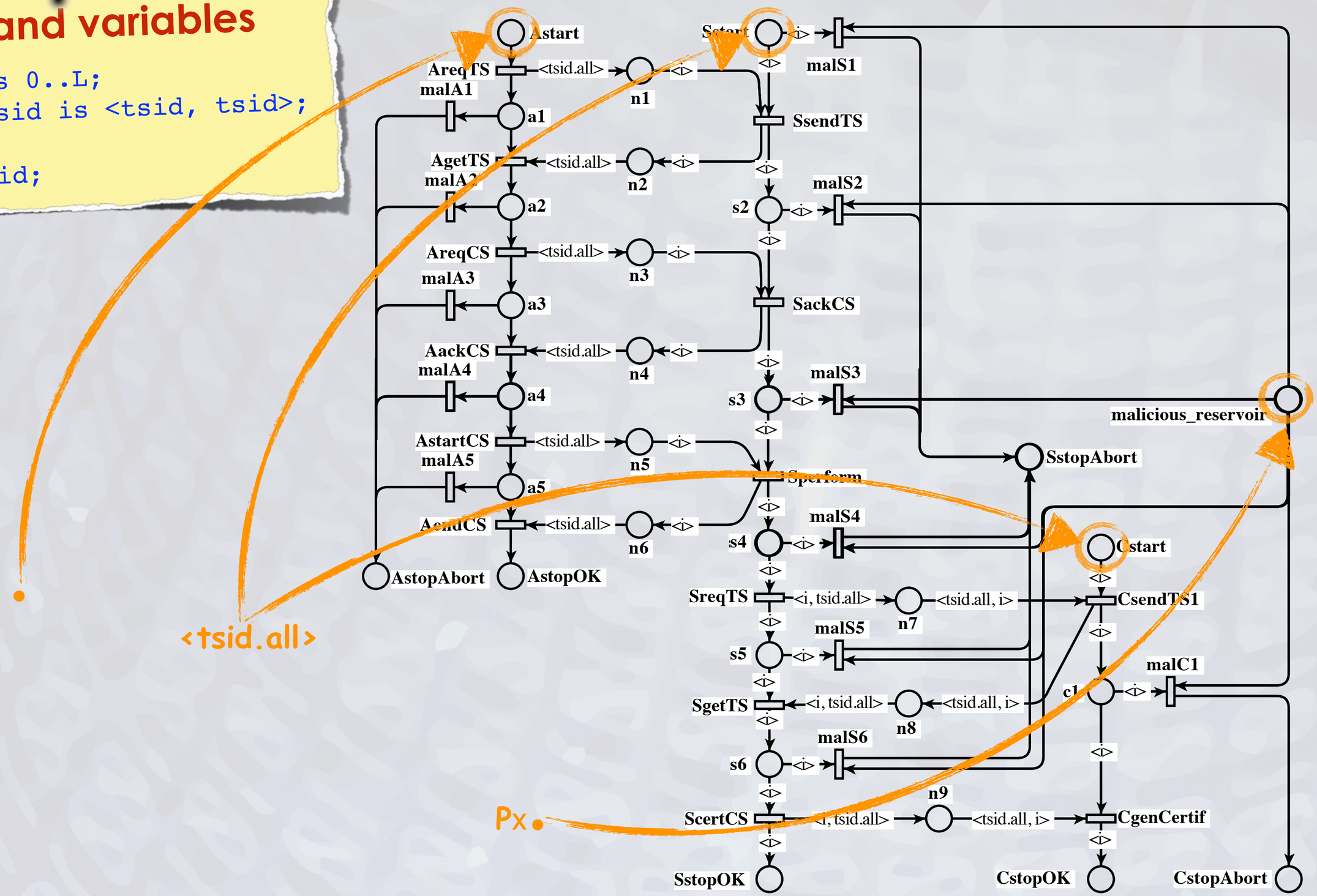
```
type tsid is 0..L;
type tsidxtsid is <tsid, tsid>;

var i in tsid;
```



Modeling the protocol (step 1)

Types and variables

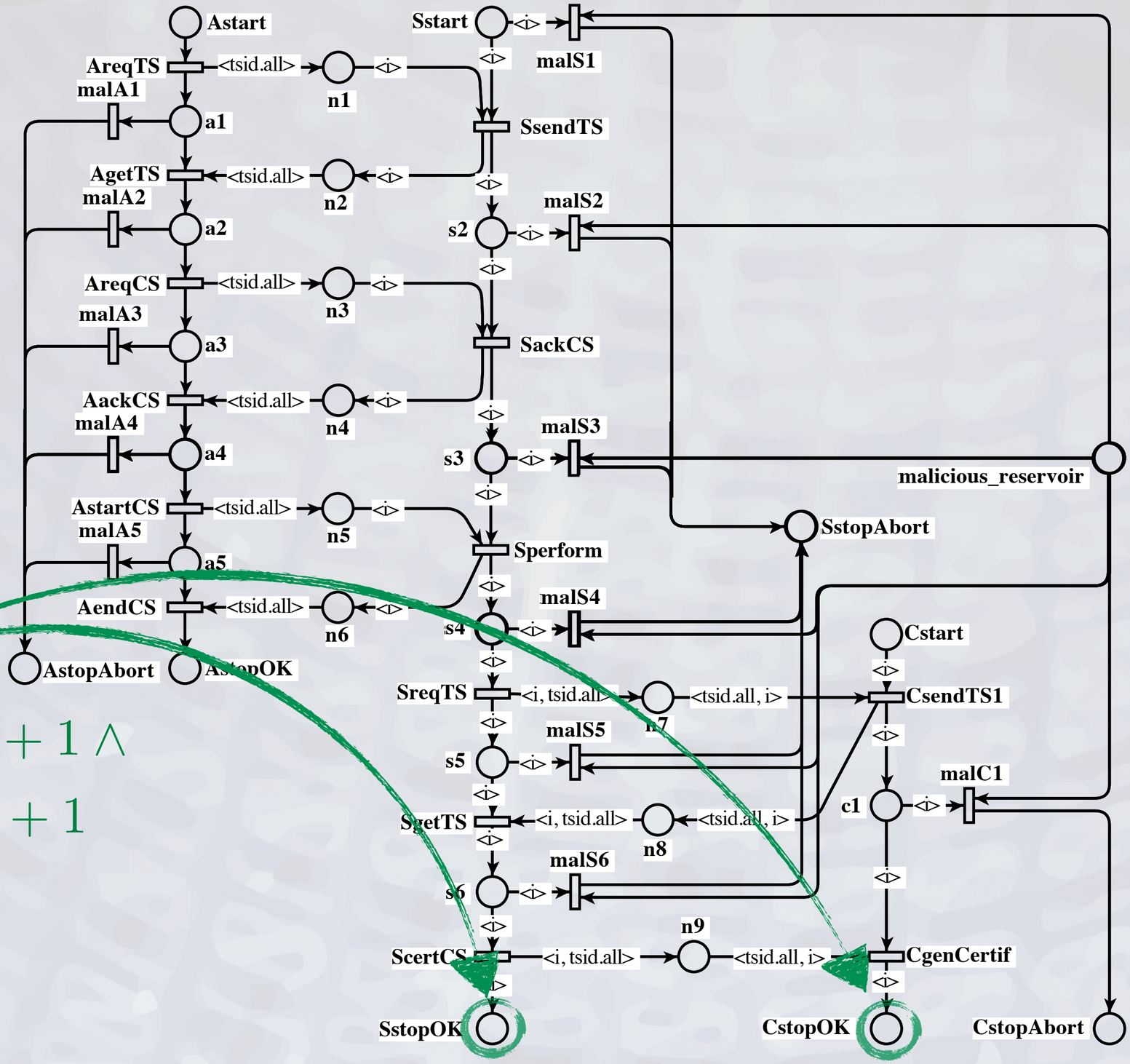
```
type tsid is 0..L;
type tsidxtsid is <tsid, tsid>;

var i in tsid;
```



Modeling the protocol (step 1)

$$F_{ok} = |SstopOK| > \frac{L}{2} \wedge |CstopOK| > \frac{L}{2}$$

$$F_{ok} = |SstopOK| = L+1 \wedge |CstopOK| = L+1$$

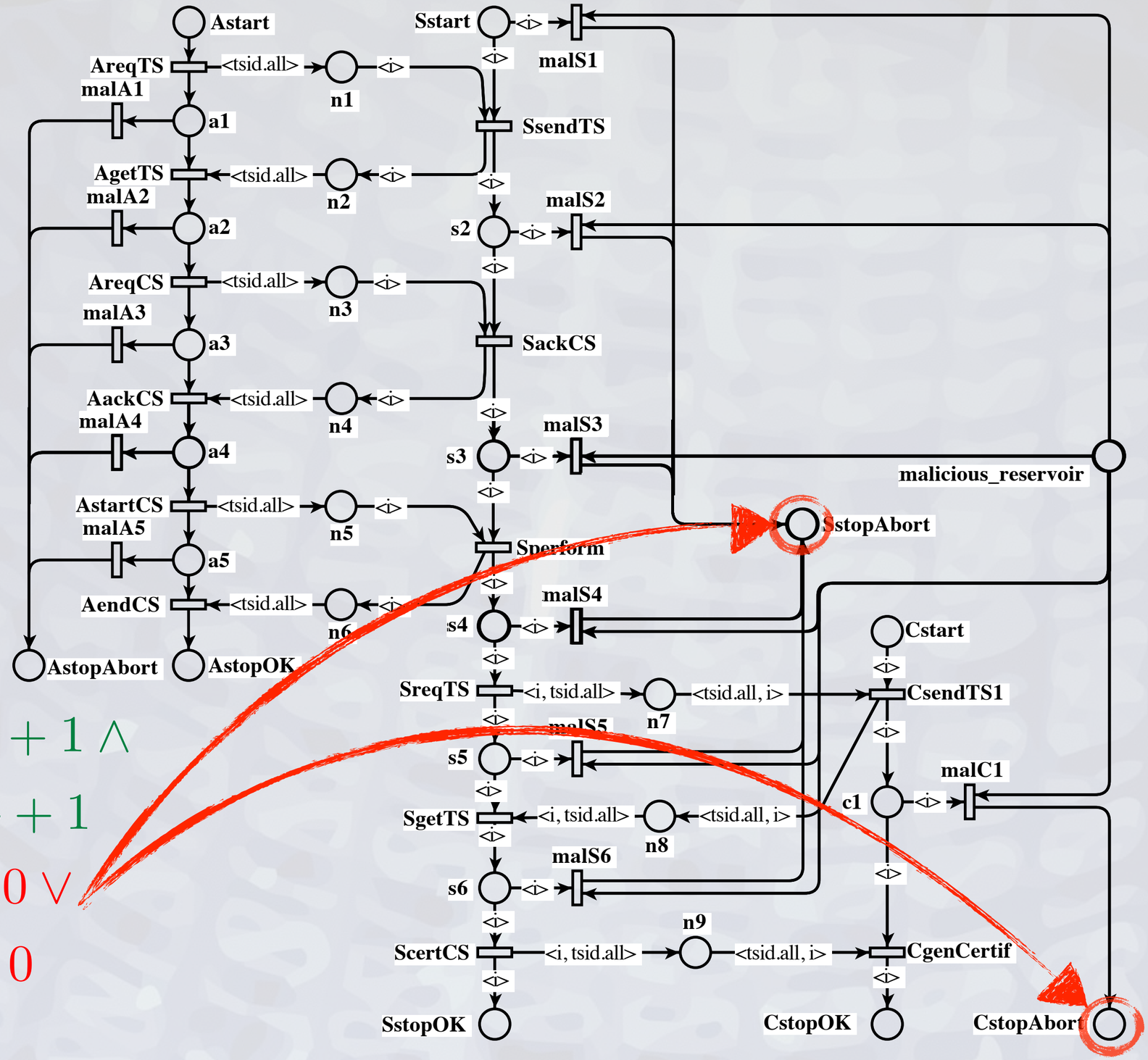


Modeling the protocol (step 1)

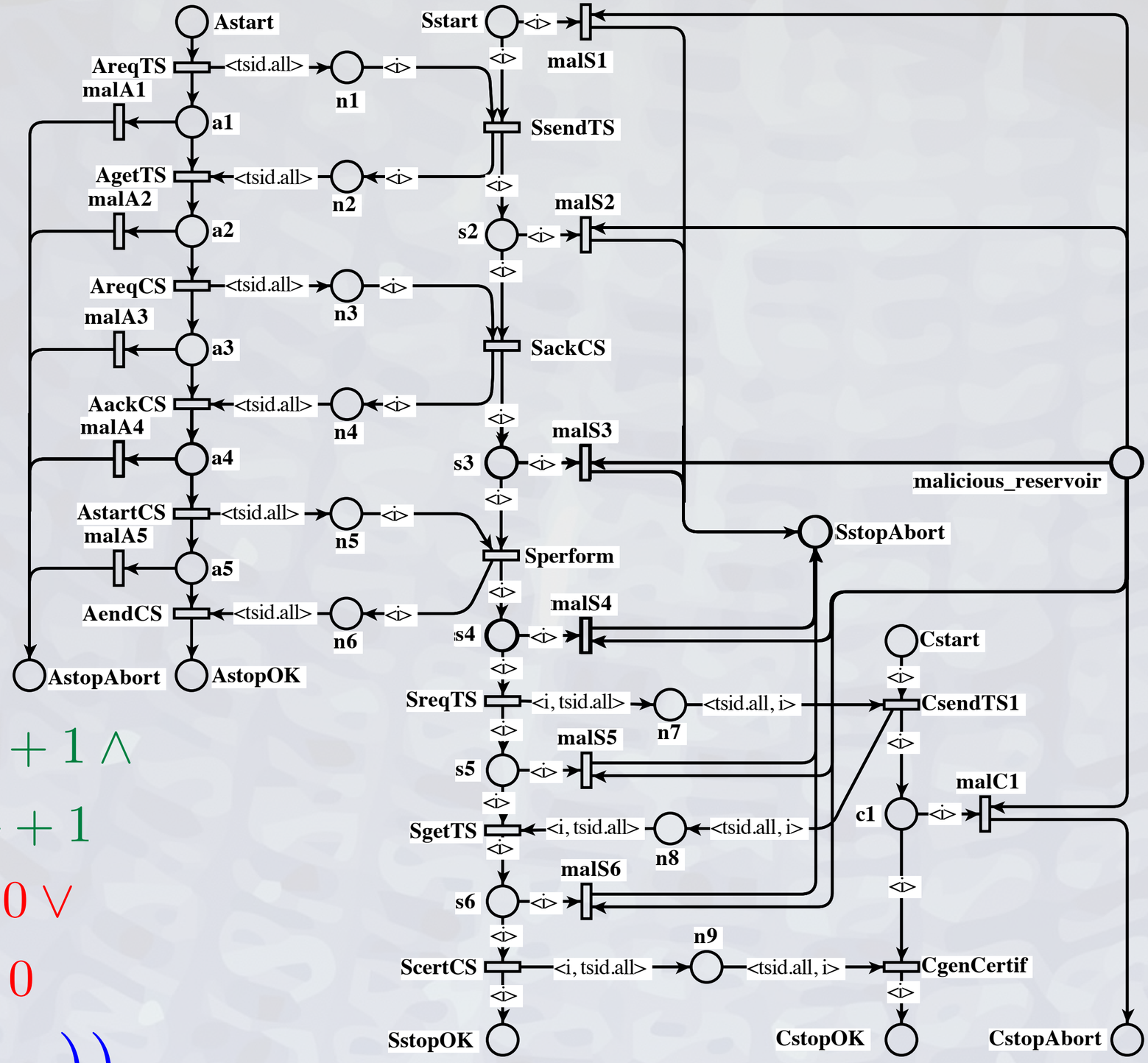
$$F_{abort} = |SstopAbort| > \frac{L}{2} \vee |CstopAbort| > \frac{L}{2}$$

$$F_{ok} = |SstopOK| = L+1 \wedge |CstopOK| = L+1$$

$$F_{abort} = |SstopAbort| > 0 \vee |CstopAbort| > 0$$



Modeling the protocol (step 1)



$$F_{ok} = |SstopOK| = L + 1 \wedge |CstopOK| = L + 1$$

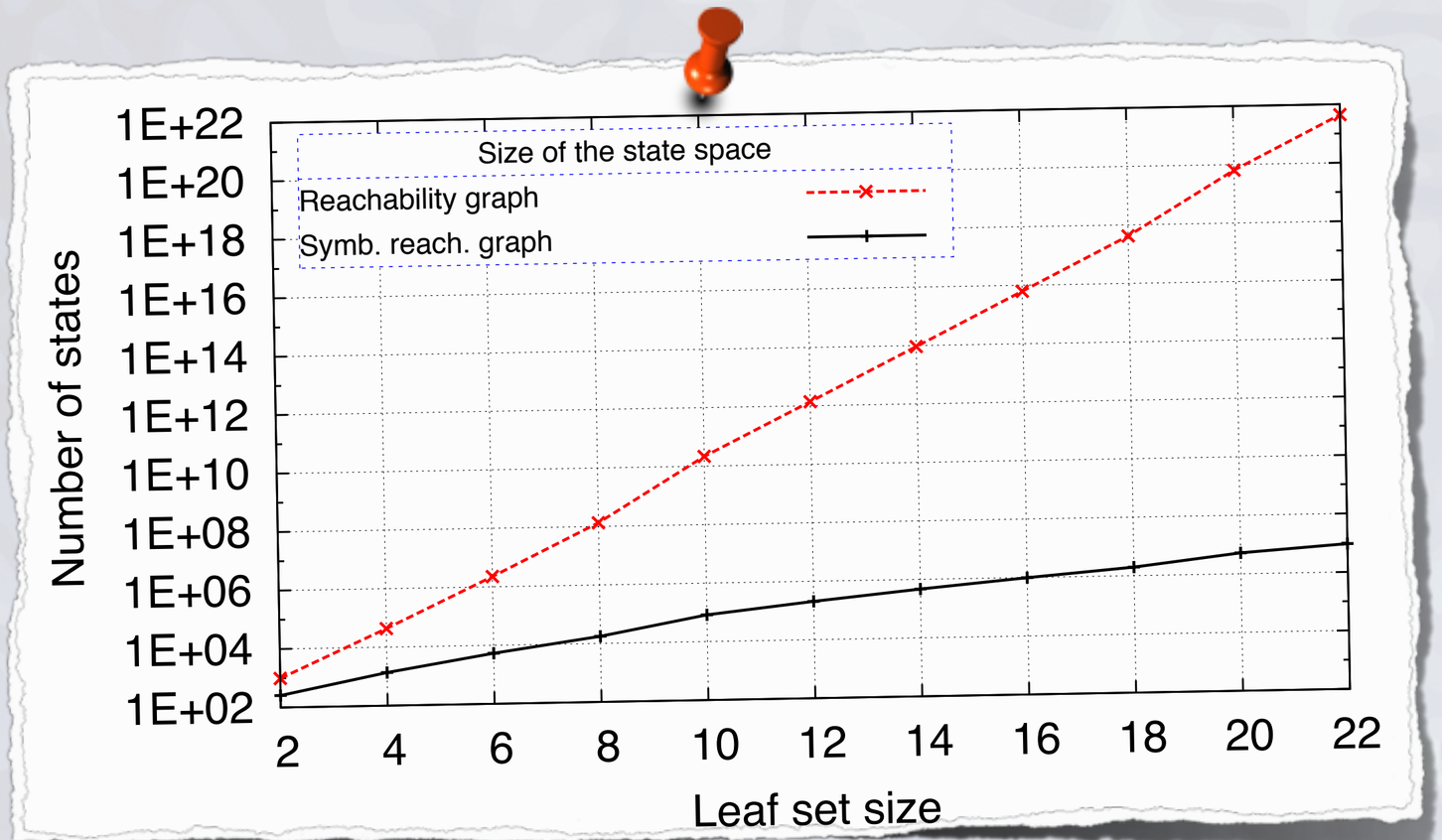
$$F_{abort} = |SstopAbort| > 0 \vee |CstopAbort| > 0$$

$$AF(F_{ok} \vee F_{abort})$$

Verifying the perfect world

About the complexity of the state space

- Roughly 10^L states
- 1 state= 13 int + 17 multistep ⇒ memory problem to check for L=32



Verifying the perfect world

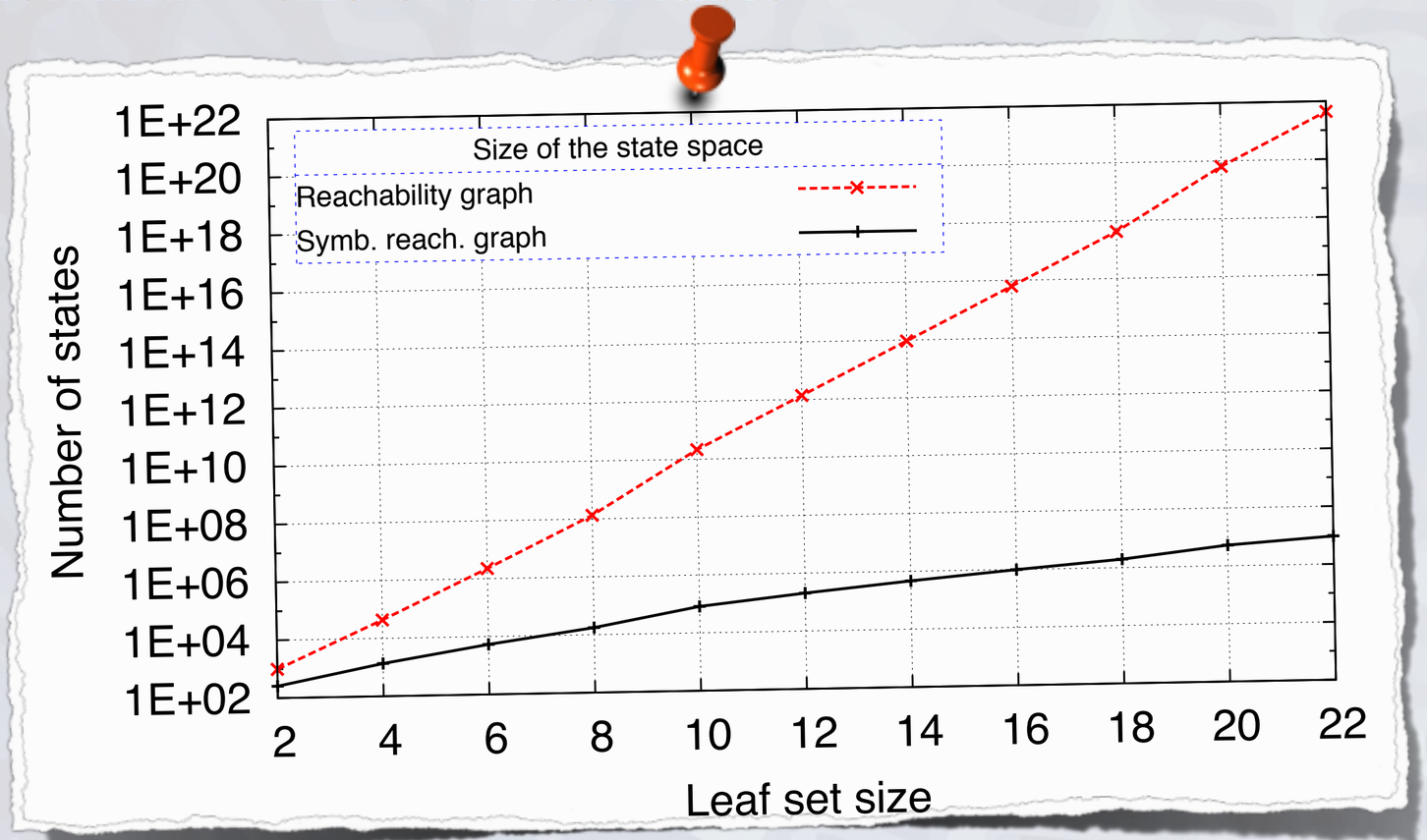
About the complexity of the state space

- Roughly 10^L states
- 1 state= 13 int + 17 multistep ⇒ memory problem to check for L=32

GreatSPN (use of symmetries)

- L=24 fails after 11h45mn of CPU (costly canonisation function)

Implementation limit = size of a hash table



Verifying the perfect world

About the complexity of the state space

- Roughly 10^L states
- 1 state= 13 int + 17 multistep ⇒ memory problem to check for L=32

GreatSPN (use of symmetries)

- L=24 fails after 11h45mn of CPU (costly canonisation function)
  - Implementation limit = size of a hash table

PNXDD (decision diagrams + variable ordering)

- Unfolding to P/T nets
- L=10 fails after 3h20mn (memory overflow > 16GB)

Verifying the perfect world

- **About the complexity of the state space**
  - Roughly 10^L states
  - 1 state= 13 int + 17 multistep ⇒ memory problem to check for L=32
- **GreatSPN (use of symmetries)**
  - L=24 fails after 11h45mn of CPU (costly canonisation function)
    - **Implementation limit = size of a hash table**
- **PNXDD (decision diagrams + variable ordering)**
  - Unfolding to P/T nets
  - L=10 fails after 3h20mn (memory overflow > 16GB)
- **ITS-Tools (hierarchical decision diagrams)**
  - Completed for L=32 in less than one minute
    - **Handling of symmetries in the system by means of a dedicated encoding**

Verifying the perfect world

http://cosyverif.org

Probabilistic analysis (step 2)

Classical approach of the domain

- Based on **p**, probability of node failure

Hypotheses required

Diversity routing to avoid coalitions

Probabilistic analysis (step 2)

Classical approach of the domain

- Based on **p**, probability of node failure

Hypotheses required

Diversity routing to avoid coalitions

Origin of problems

- Source 1 → failure of the protocol
  - **No answer to A + no ack to A**
  - **Interactions between A, S (leafset)**
- Source 2 → inappropriate certificate
  - **Lost of a certificate (inconsistency)**
  - **Interactions between S (leafset) and C (leafset)**

Probabilistic analysis (step 2)

Classical approach of the domain

- Based on **p**, probability of node failure

Hypotheses required

Diversity routing to avoid coalitions

Numerical applications

p = 0.3 («untrusted»)

P = 0.05 («trusted»)

Origin of problems

- Source 1 → failure of the protocol
  - **No answer to A + no ack to A**
  - **Interactions between A, S (leafset)**
- Source 2 → inappropriate certificate
  - **Lost of a certificate (inconsistency)**
  - **Interactions between S (leafset) and C (leafset)**

Formulas and experimental values

Protocol failure

- More that L/2 nodes are malicious
- The formula:

$$\underbrace{\sum_{i=1}^{L+1} \binom{L+1}{i} p^i (1-p)^{L+1-i}}_{\text{at most L+1 malicious nodes}} - \underbrace{\sum_{i=1}^{\frac{L}{2}} \binom{L+1}{i} p^i (1-p)^{L+1-i}}_{\text{at most L/2 malicious nodes}}$$

Formulas and experimental values

Protocol failure

- More that L/2 nodes are malicious
- The formula:

$$\underbrace{\sum_{i=1}^{L+1} \binom{L+1}{i} p^i (1-p)^{L+1-i}}_{\text{at most L+1 malicious nodes}} - \underbrace{\sum_{i=1}^{\frac{L}{2}} \binom{L+1}{i} p^i (1-p)^{L+1-i}}_{\text{at most L/2 malicious nodes}}$$

L	DHT - $p = 0.3$	CORPS - $p = 0.05$
8	0.188	6.64×10^{-5}
16	0.079	6.57×10^{-8}
32	0.016	8.24×10^{-14}

Formulas and experimental values

Protocol failure

- More that L/2 nodes are malicious
- The formula:

$$\underbrace{\sum_{i=1}^{L+1} \binom{L+1}{i} p^i (1-p)^{L+1-i}}_{\text{at most L+1 malicious nodes}} - \underbrace{\sum_{i=1}^{\frac{L}{2}} \binom{L+1}{i} p^i (1-p)^{L+1-i}}_{\text{at most L/2 malicious nodes}}$$

L	DHT - $p = 0.3$	CORPS - $p = 0.05$
8	0.188	6.64×10^{-5}
16	0.079	6.57×10^{-8}
32	0.016	8.24×10^{-14}

Inappropriate certificate generation

- Two parts
  - **Existence of more L/2 malicious nodes**
  - **Unable to retrieve at least L/2 + 1 identical answers**
- The formula (combines problems between S and C)

$$1 - (1 - P_{>\frac{L}{2}})^2$$

Formulas and experimental values

Protocol failure

- More that L/2 nodes are malicious
- The formula:

$$\underbrace{\sum_{i=1}^{L+1}\binom{L+1}{i}p^i(1-p)^{L+1-i}}_{\text{at most L+1 malicious nodes}} - \underbrace{\sum_{i=1}^{\frac{L}{2}}\binom{L+1}{i}p^i(1-p)^{L+1-i}}_{\text{at most L/2 malicious nodes}}$$

L	DHT - $p = 0.3$	CORPS - $p = 0.05$
8	0.188	6.64×10^{-5}
16	0.079	6.57×10^{-8}
32	0.016	8.24×10^{-14}

Inappropriate certificate generation

- Two parts
  - **Existence of more L/2 malicious nodes**
  - **Unable to retrieve at least L/2 + 1 identical answers**
- The formula (combines problems between S and C)

$$1 - (1 - P_{>\frac{L}{2}})^2$$

$$1 - \left(1 - \sum_{i=1}^{L+1}\binom{L+1}{i}p^i(1-p)^{L+1-i} + \sum_{i=1}^{\frac{L}{2}}\binom{L+1}{i}p^i(1-p)^{L+1-i}\right)^2$$

Formulas and experimental values

Protocol failure

- More that L/2 nodes are malicious
- The formula:

$$\underbrace{\sum_{i=1}^{L+1} \binom{L+1}{i} p^i (1-p)^{L+1-i}}_{\text{at most L+1 malicious nodes}} - \underbrace{\sum_{i=1}^{\frac{L}{2}} \binom{L+1}{i} p^i (1-p)^{L+1-i}}_{\text{at most L/2 malicious nodes}}$$

L	DHT - $p = 0.3$	CORPS - $p = 0.05$
8	0.188	6.64×10^{-5}
16	0.079	6.57×10^{-8}
32	0.016	8.24×10^{-14}

Inappropriate certificate generation

- Two parts
  - **Existence of more L/2 malicious nodes**
  - **Unable to retrieve at least L/2 + 1 identic**
- The formula (combines problems b

$$1 - (1 - P_{>\frac{L}{2}})^2$$

$$1 - \left(1 - \sum_{i=1}^{L+1} \binom{L+1}{i} p^i (1-p)^{L+1-i} + \sum_{i=1}^{\frac{L}{2}} \binom{L+1}{i} p^i (1-p)^{L+1-i}\right)^2$$

L	DHT - $p = 0.3$	CORPS - $p = 0.05$
8	0.216	4.97×10^{-4}
16	0.075	3.50×10^{-8}
32	0.014	4.24×10^{-13}

Conclusion

- **Quasi-certification entity (elaboration + verification)**
  - Low probability of failure + Good message complexity (not discussed)

Conclusion

Quasi-certification entity (elaboration + verification)

- Low probability of failure + Good message complexity (not discussed)
- Application to digital tax filling in France
  - **36.5 M revenues declarations (in 2012)**
  - **a wrong tax certificate every 5132 years**
  - **lost of a tax certificate every 997 years**

Conclusion

Quasi-certification entity (elaboration + verification)

- Low probability of failure + Good message complexity (not discussed)
- Application to digital tax filling in France
  - **36.5 M revenues declarations (in 2012)**
  - **a wrong tax certificate every 5132 years**
  - **lost of a tax certificate every 997 years**

3 years of work (details recently fixed)

- **Work partially published in TrustCom'2013**
- **Without formal proof (there was yet glitches details with hypotheses)**

Conclusion

- **Quasi-certification entity (elaboration + verification)**
  - Low probability of failure + Good message complexity (not discussed)
  - Application to digital tax filling in France
    - **36.5 M revenues declarations (in 2012)**
    - **a wrong tax certificate every 5132 years**
    - **lost of a tax certificate every 997 years**
- **3 years of work (details recently fixed)**
  - **Work partially published in TrustCom'2013**
  - **Without formal proof (there was yet glitches details with hypotheses)**
- **Realistic problem with applicability to e-government**
  - Probably numerous applications in the future
  - The model is now a metric for the Model Checking Contest
  - Potential applicability for Symmetric Nets with Bags
    - **Excellent playground for this type of problems**