Towards verification of Bluetooth device discovery

Marie Duflot

Outline

- Description of the protocol
- Modelling
- First results
- We want more ....

Description of the protocol

Protocol overview

- short-range low-power wireless protocol
- frequency hopping over 79/32 frequencies

Protocol overview

- short-range low-power wireless protocol
- frequency hopping over 79/32 frequencies
  - ➤ need to form piconets
    - processes know when to send/receive
    - processes know the hopping frequency
    - master-slave roles

Protocol overview

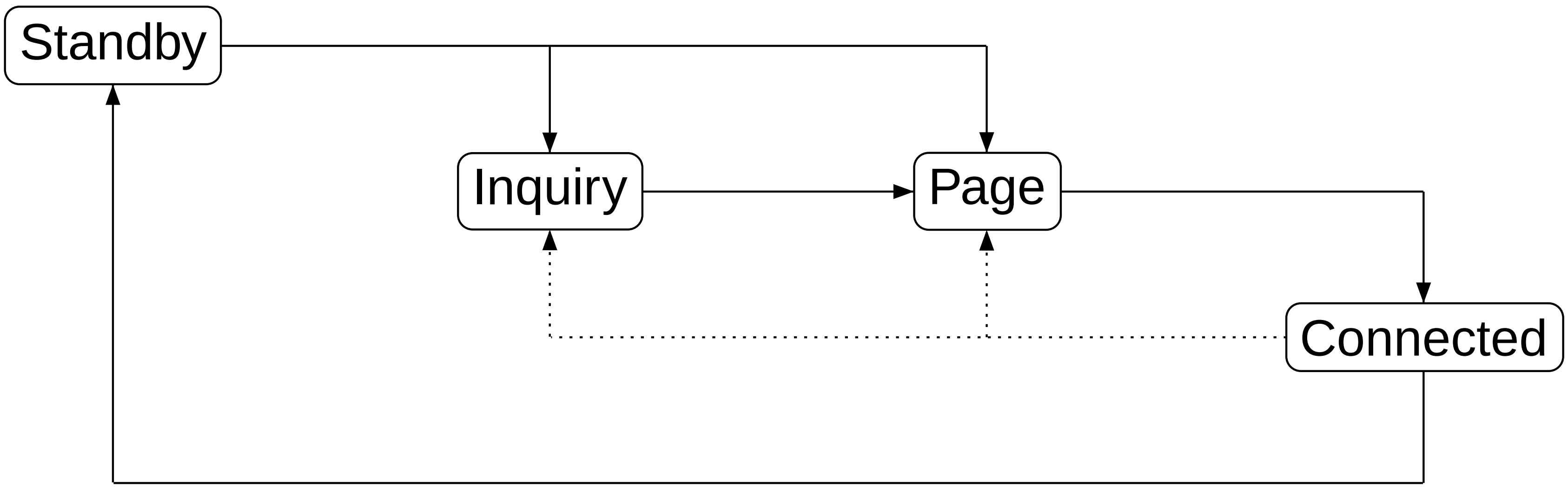
- short-range low-power wireless protocol
- frequency hopping over 79/32 frequencies
  - ➤ need to form piconets
    - processes know when to send/receive
    - processes know the hopping frequency
    - master-slave roles
  - ➤ no communication before initialisation

Protocol overview

- short-range low-power wireless protocol
- frequency hopping over 79/32 frequencies
  - ➢ need to form piconets
    - processes know when to send/receive
    - processes know the hopping frequency
    - master-slave roles
  - ➢ no communication before initialisation

First mandatory step: device discovery

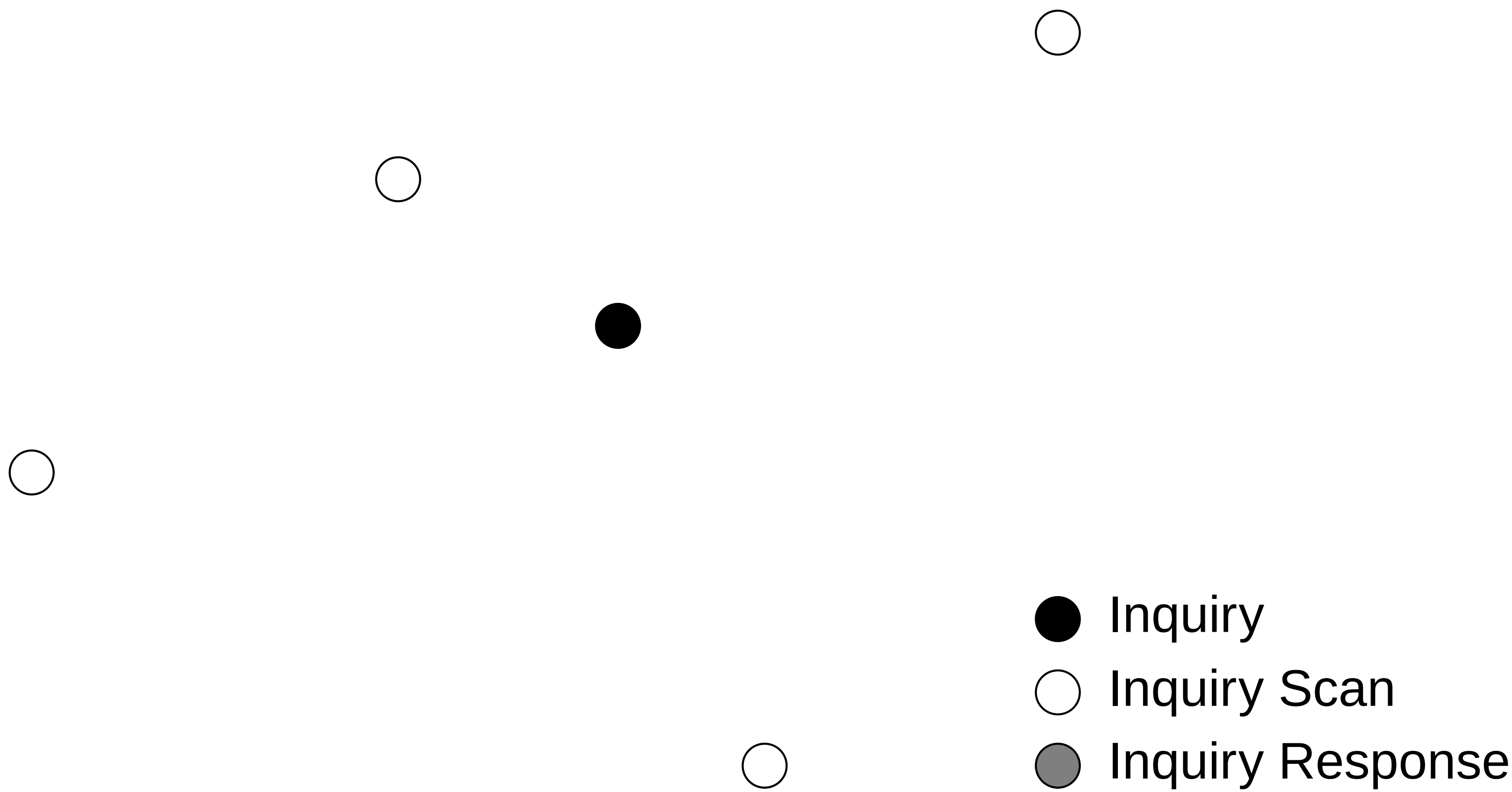
States of a Bluetooth device



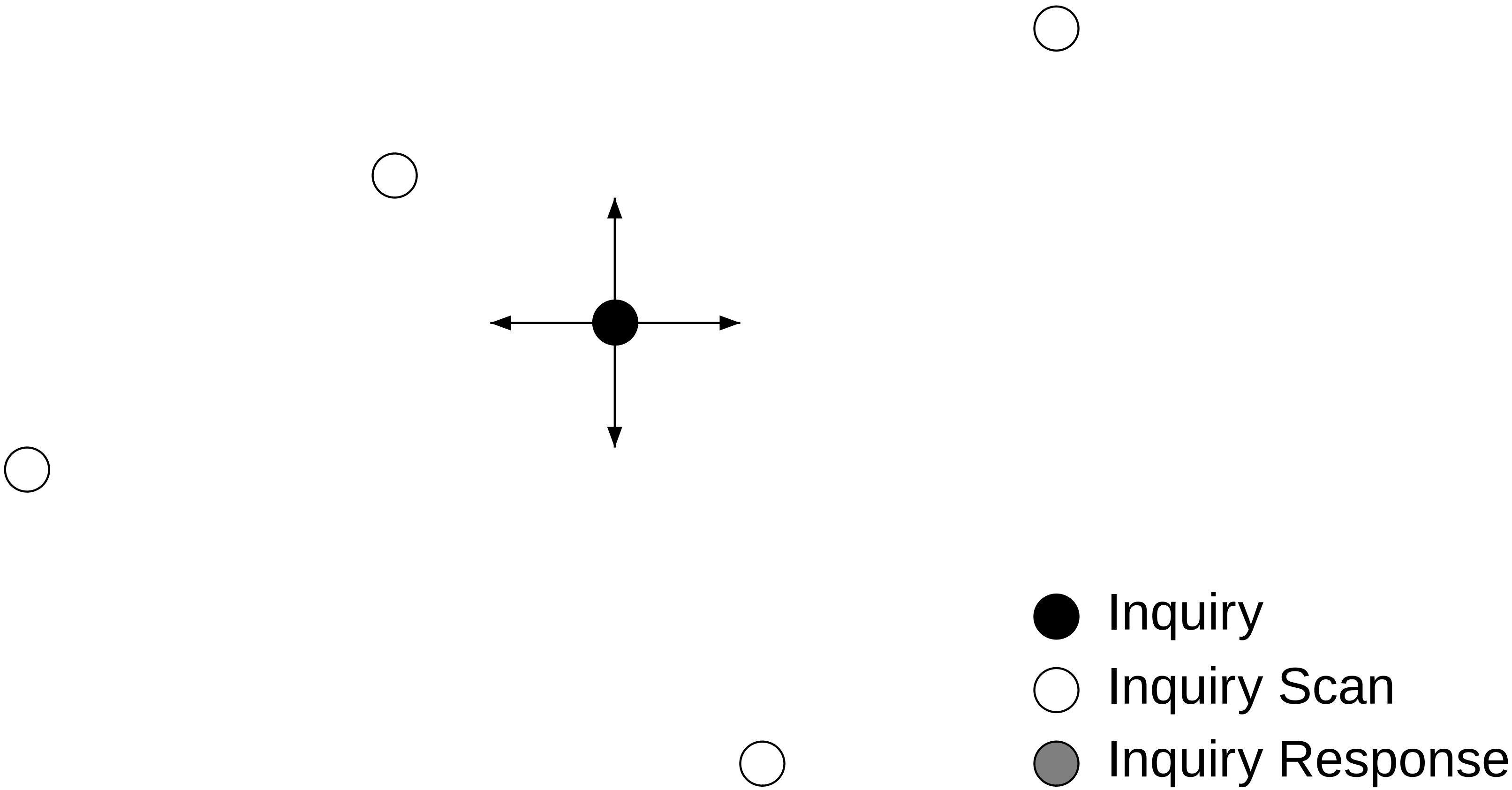
Standby: default operational state

Connected: device ready to communicate in a piconet

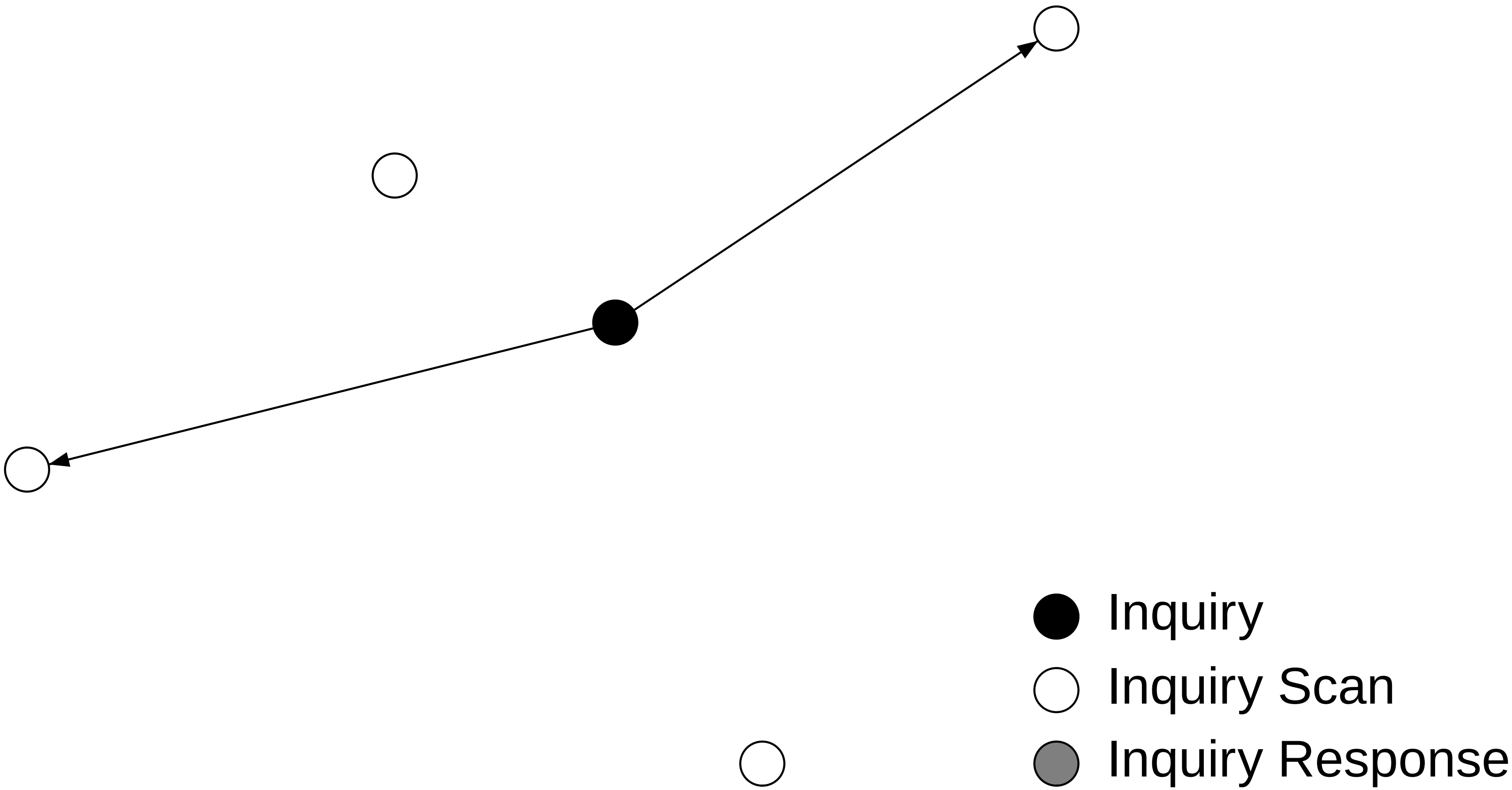
Inquiry (version 1.2)



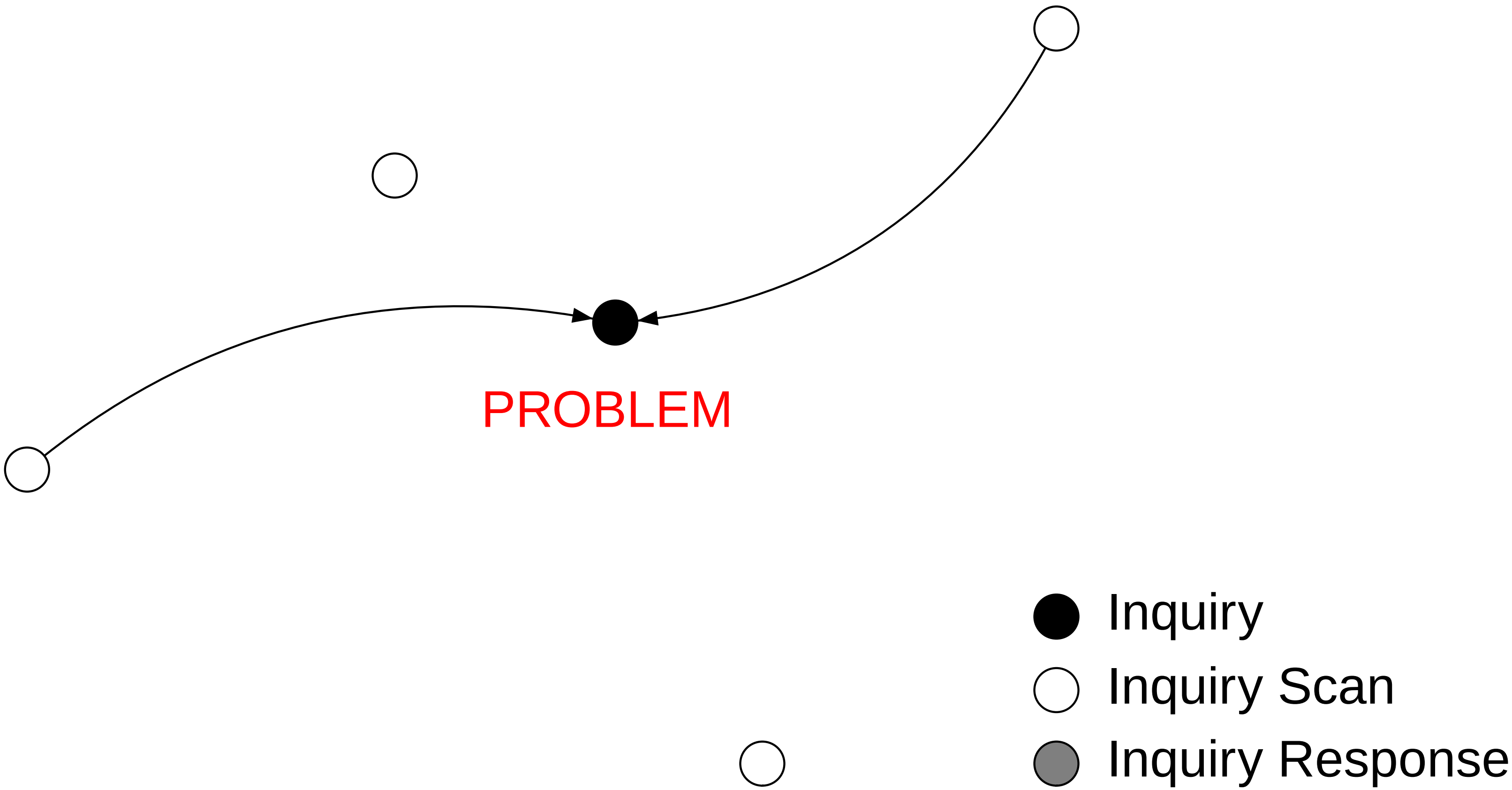
Inquiry (version 1.2)



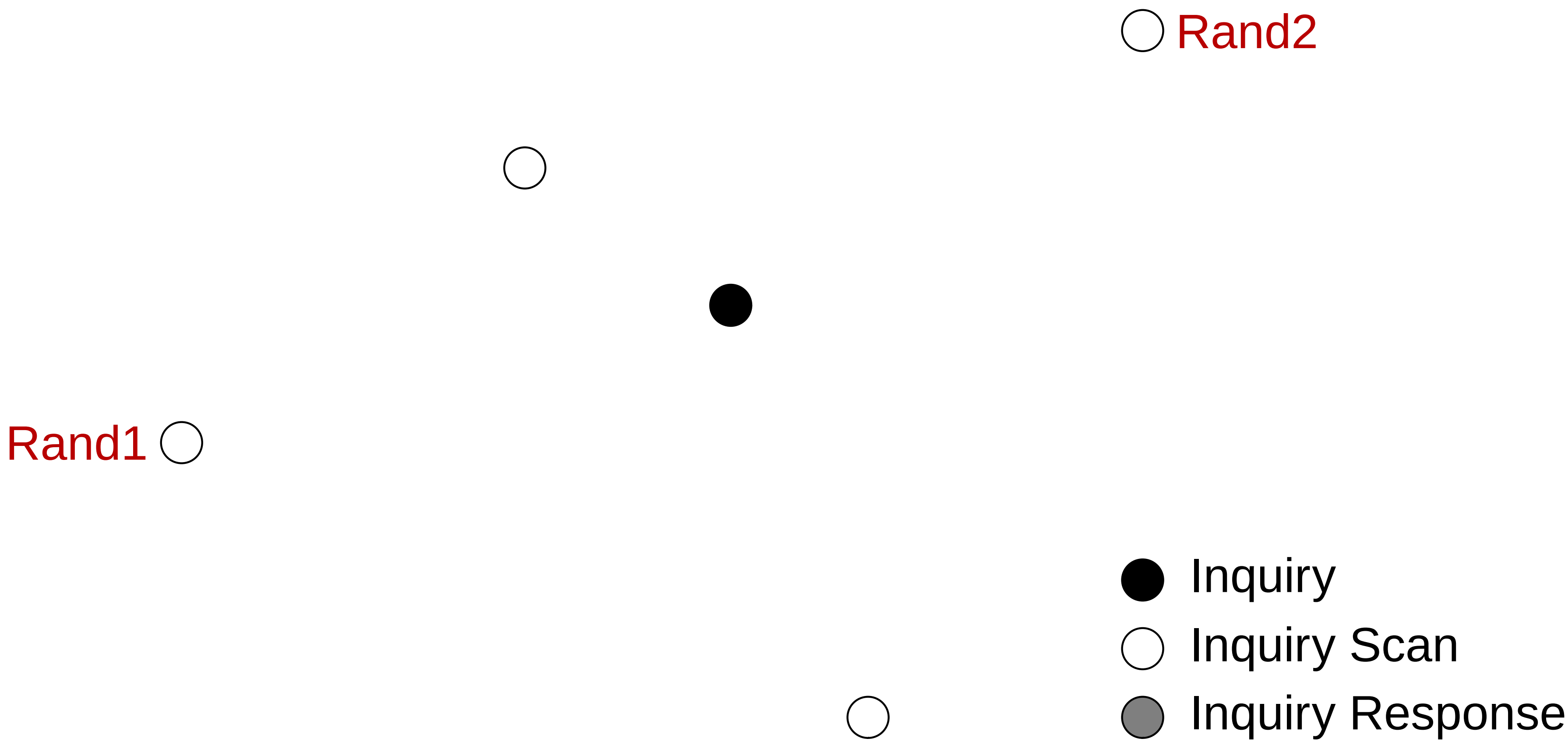
Inquiry (version 1.2)



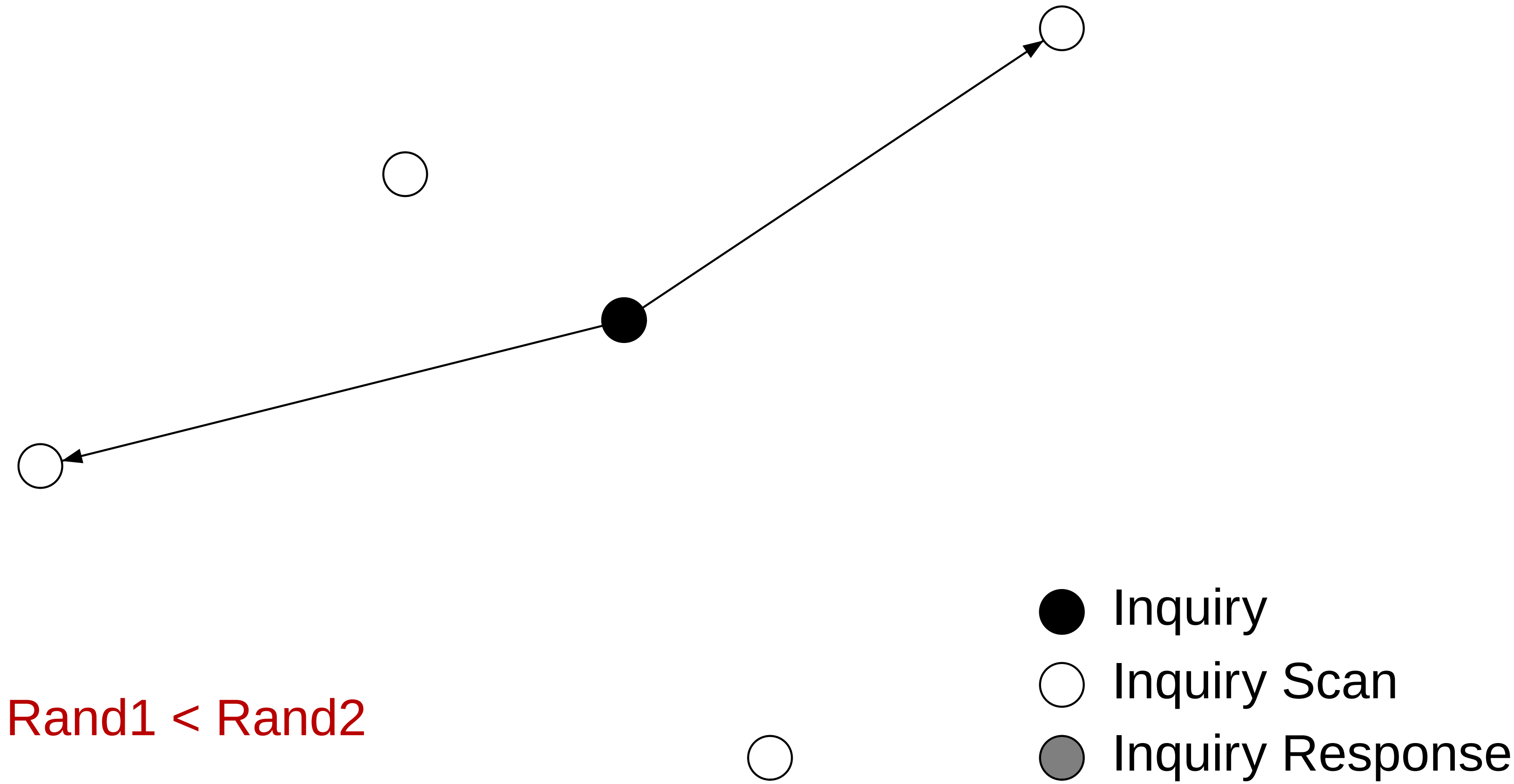
Inquiry (version 1.2)



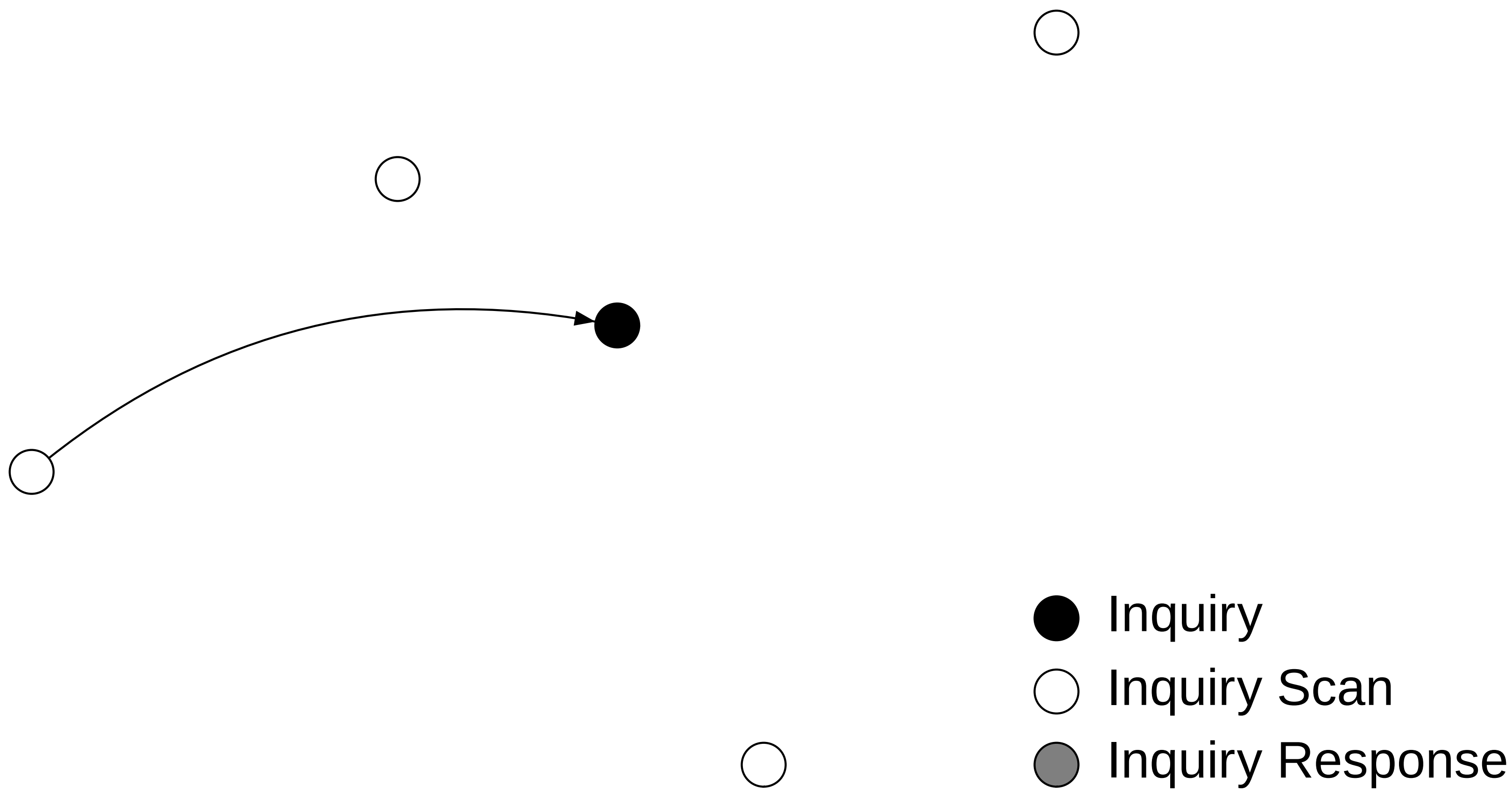
Inquiry (version 1.2)



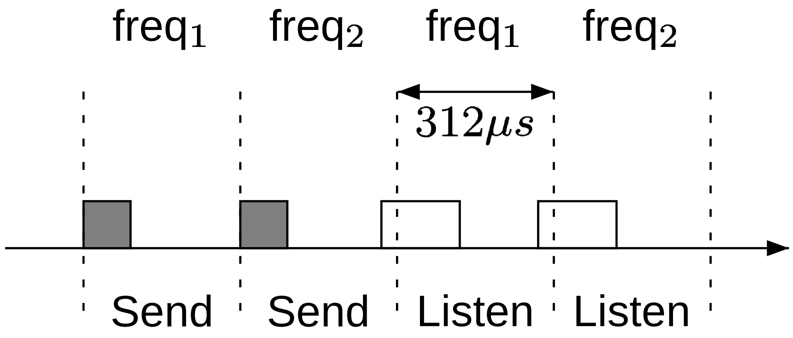
Inquiry (version 1.2)



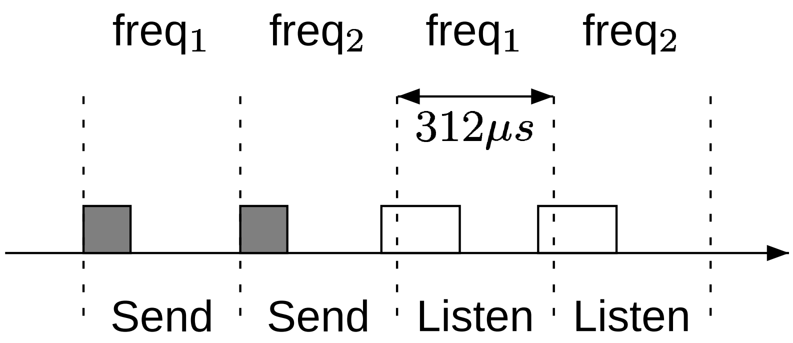
Inquiry (version 1.2)



The sender

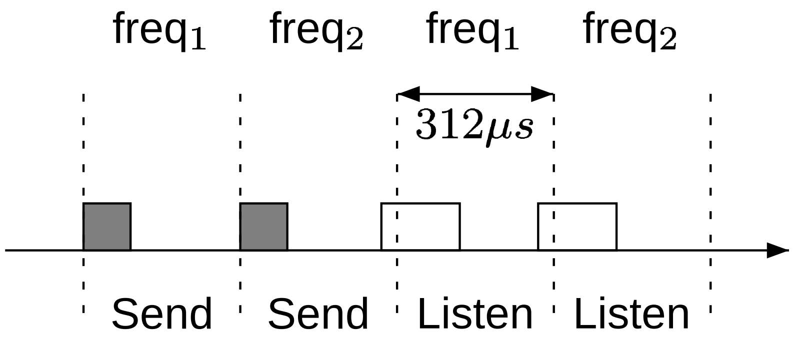


The sender



freq $= [CLK_{16-12} + k$
$+ (CLK_{4-2,0} - CLK_{16-12})$ mod $16]$ mod 32

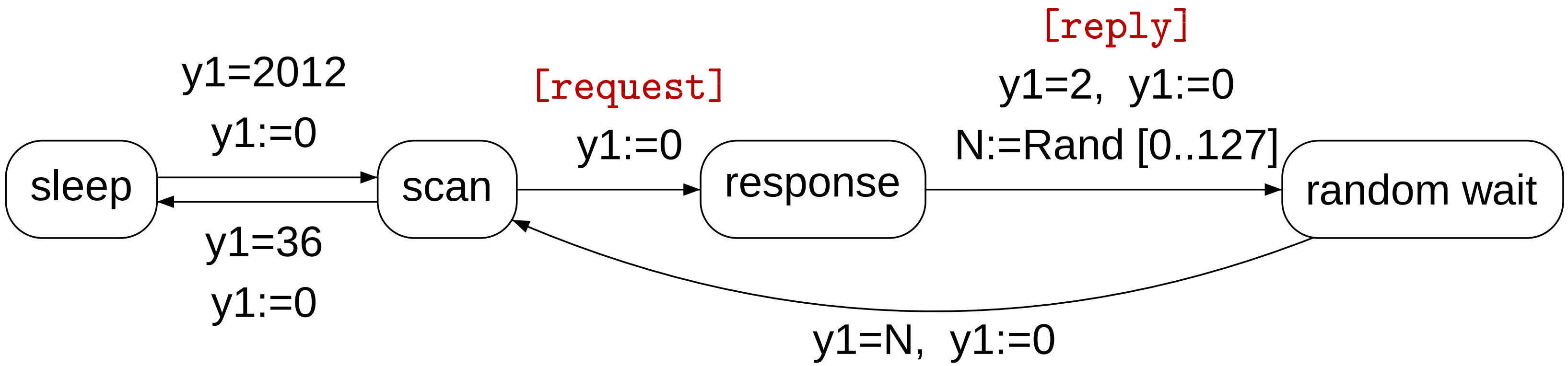
The sender



freq $= [CLK_{16-12} + k$
$+ (CLK_{4-2,0} - CLK_{16-12})$ mod $16]$ mod 32

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	2	3	20	21	22	23	24	25	26	27	28	29	30	31	32
17	18	19	20	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	23	24	25	26	27	28	29	30	31	32
1	2	3	4	5	6	7	24	25	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	27	28	29	30	31	32
1	2	3	4	5	6	7	8	9	10	11	28	29	30	31	32
17	18	19	20	21	22	23	24	25	26	27	28	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	31	32
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	32
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
17	18	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	21	22	23	24	25	26	27	28	29	30	31	32
1	2	3	4	5	22	23	24	25	26	27	28	29	30	31	32
17	18	19	20	21	22	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	25	26	27	28	29	30	31	32
1	2	3	4	5	6	7	8	9	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24	25	26	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	29	30	31	32
1	2	3	4	5	6	7	8	9	10	11	12	13	30	31	32
17	18	19	20	21	22	23	24	25	26	27	28	29	30	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	16

The receiver



- [request]: message sent by the sender
- [reply]: message sent by the receiver
- need to compute the frequency at which the receiver is listening (phase)
- phase increased by one each time the receiver replies

Modelling the protocol

Modelling formalism

- randomised back-off
  - ➤ need probabilistic model
- discrete time slots
- no nondeterminism
  - ➤ no nondeterministic choice within a device
  - ➤ full synchronisation between devices

➜ discrete-time Markov Chains (DTMCs)

Constants from Bluetooth standard

- Sender changes state every time slot
- Receiver can wait for 2012 time slots without changing state
- 2 trains of 16 frequencies
- The trains change with time
- A train is repeated 256 times before switching
- The phase changes every 4096 slots
  - ➤ A Huge model
  - ➤ Too many possible initial states

Receiver's frequencies

```
module frequency1

  z1 : [1..phase];// clock for phase
  f1 : [1..16]; // frequency of receiver
  o1 : [0..1]; // offset of receiver

  // update frequency (1 slot passes)
  [time] z1<phase -> (z1'=z1+1);
  [time] z1=phase -> (z1'=1) & (f1'=f1<16?f1+1:1)& (o1'=f1<16?o1:1-o1);
  // update frequency: something is sent by the receiver
  [reply] true -> (f1'=(f1<16)?f1+1:1) & (o1'=(f1<16)?o1:1-o1);

endmodule
```

Abstractions (1): the sender

```
[time] (x=0) -> (x'=1) ;
[](x=1) & (send=1) -> (send'=2) ;
```

Abstractions (1): the sender

```
[time] (x=0) -> (x'=1) ;
[](x=1) & (send=1) -> (send'=2) ;
```

$$\Downarrow$$

```
[time] (send=1) -> (send'=2) ;
```

Abstractions (1): the sender

```
[time] (x=0) -> (x'=1) ;
[](x=1) & (send=1) -> (send'=2) ;
```

$$\Downarrow$$

```
[time] (send=1) -> (send'=2) ;
```

- no clock for the sender
- sender totally deterministic

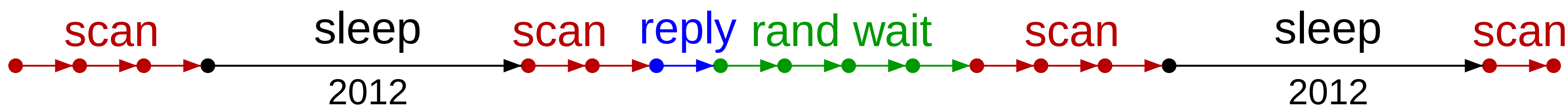
Abstractions (2): the receiver

- Initial execution:

Abstractions (2): the receiver

- Initial execution:

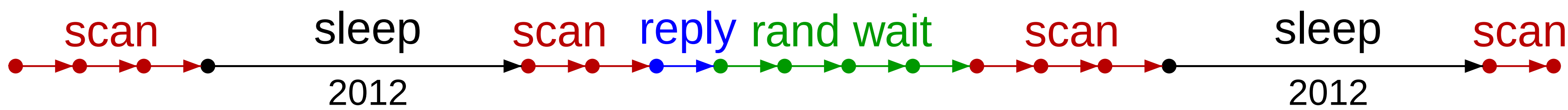
- Merge consecutive sleep transitions into one



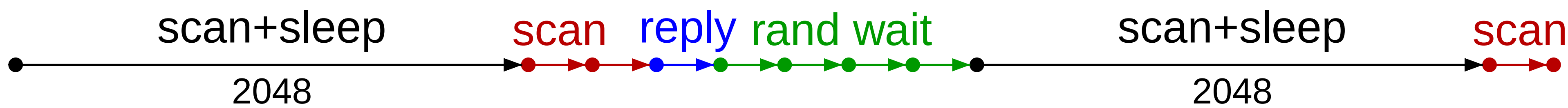
Abstractions (2): the receiver

- Initial execution:

- Merge consecutive sleep transitions into one



- Formula `success`: decides if the receiver is going to hear something.
  - ➤ Merge scan with sleep when hears nothing



➜ works only for one receiver

Abstractions (3): starting point

Abstractions (3): starting point

- need to fix a scenario
- sender's state entirely determined by its clock
  - ➤ doesn't start inquiring in a precise state
- receiver's point of view
  - ➤ a sender is already inquiring
  - ➤ we start when the receiver scans

Abstractions (3): starting point

- need to fix a scenario
- sender's state entirely determined by its clock
  - ➤ doesn't start inquiring in a precise state
- receiver's point of view
  - ➤ a sender is already inquiring
  - ➤ we start when the receiver scans

But ... still 17 thousand million initial states

Verification and Results

Verification with PRISM

Model checking *vs.* simulation

- examine lower-level detail
- worst case, not only average

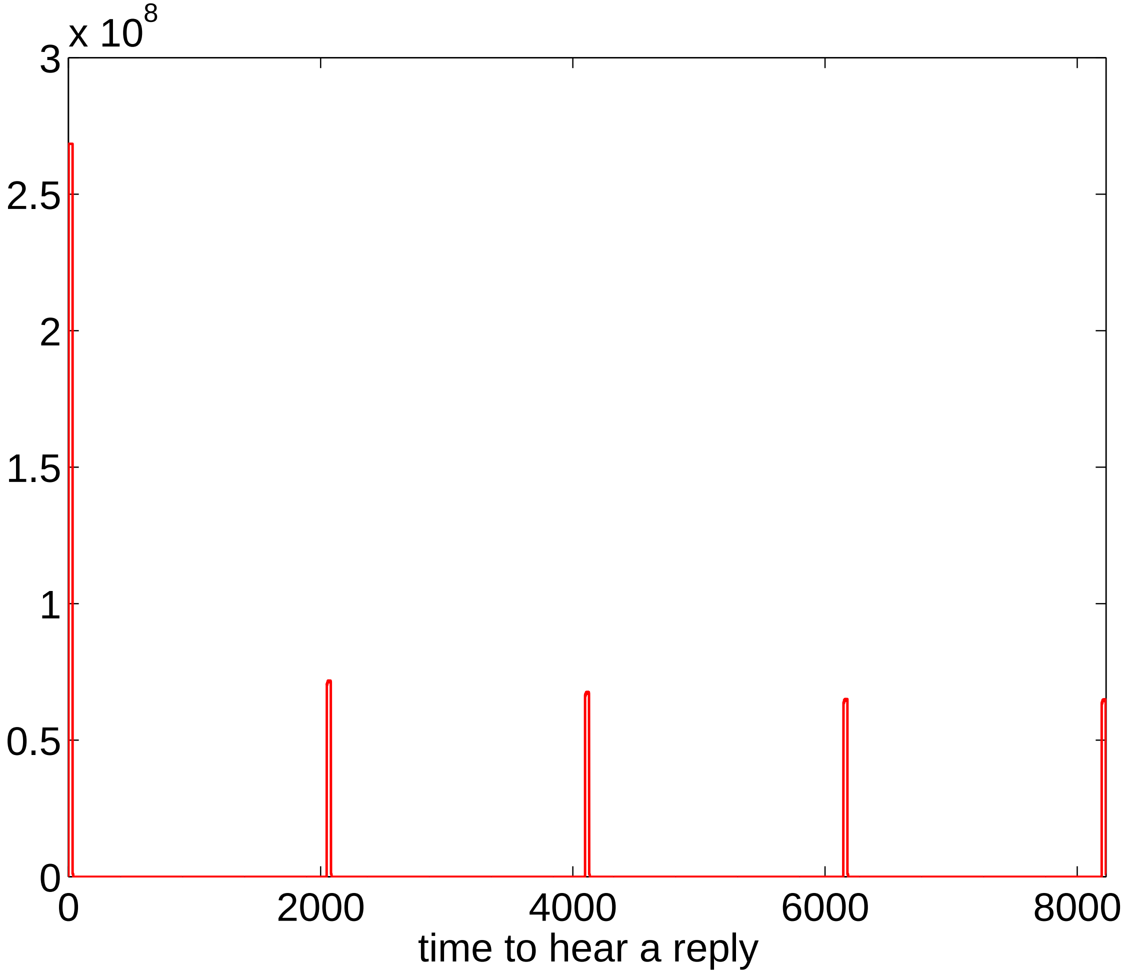
New version of PRISM including:

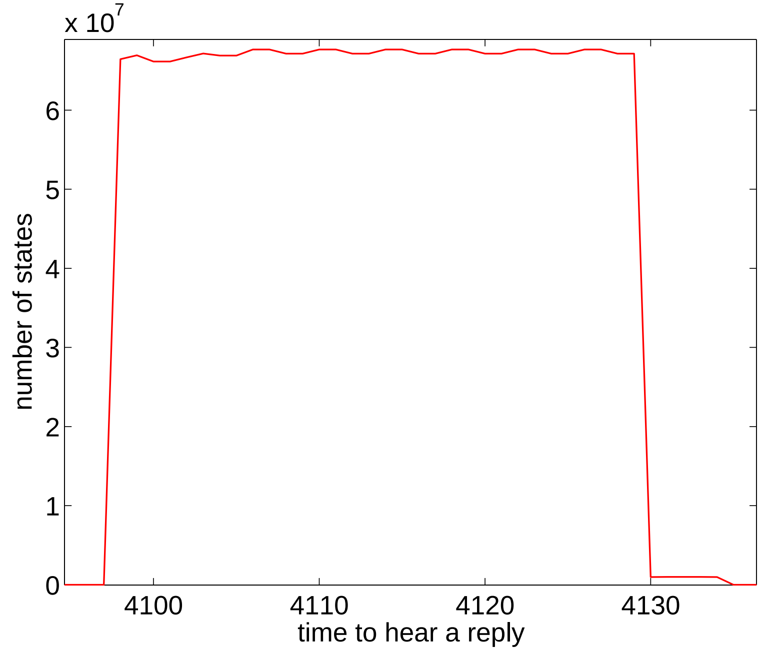
- computation of expected time
  `[label,cost] condition -> update`
  `R=?  [F rec=2]`
- multiple initial states
  Predicate over variables *vs.* a value for each variable
- additional queries for results over many states
  `R=?  [F rec=2 {"init"}{min}{max}]`

Expected time to receive one reply

- Very big models $\rightarrow$ symbolic implementation (MTBDDs)
- Initial states split into 32 classes (possible initial frequencies)
- 32 models of around 3 thousand million states each
- 55-57 seconds to build one model
- 3-4 seconds to check the property

Graph of the results





Analysis of the results

- Time to reply to one message: min 2, max 8,229 slots (2.5 seconds)
- probability of replying to message after sleeping N times

N	0	1	2	3	4
p	0.5003	0.6335	0.7591	0.8797	1

- Analysis of the worst case expected time

1	2	3	20	21	22	23	24	25	26	27	28	29	30	31	32
17	18	19	20	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	23	24	25	26	27	28	29	30	31	32

  - ➤ sender starts on frequency 3
  - ➤ receiver starts on frequency 2
  - ➤ last repetition of the train

Expected time to receive two replies

- Size of the models: 51 thousand million states
- Time to build the models: 30 minutes
- Time to check the property: 80 minutes
- Maximum expected time: 16,502 slots (5 seconds)

It's not over yet!!!

- Count up to more replies
- Consider more than one receiver
- Compare version 1.1 and 1.2

It's not over yet!!!

- Count up to more replies
- Consider more than one receiver
- Compare version 1.1 and 1.2

Problem:

State space!!!

It's not over yet!!!

- Count up to more replies
- Consider more than one receiver
- Compare version 1.1 and 1.2

Problem:

State space!!!

Solutions:

- Simulation?
- Scaling?
- Abstraction?