Probabilistic Model Checking

Marta Kwiatkowska
Gethin Norman
Dave Parker

University of Oxford

Part 3 – DTMC Case Studies

Overview

- Introduce two real-world examples
  - derived models as discrete-time Markov chains
  - quantitatively analysed them (with PRISM)
  - observed unusual trends…
- Bluetooth® device discovery
  - worked from the standard document (1000 pages), versions 1.1 and 1.2
- Contract signing
  - worked from the original paper, discovered a flaw and proposed a fix
- See PRISM webpage for models and more analysis…

Bluetooth device discovery

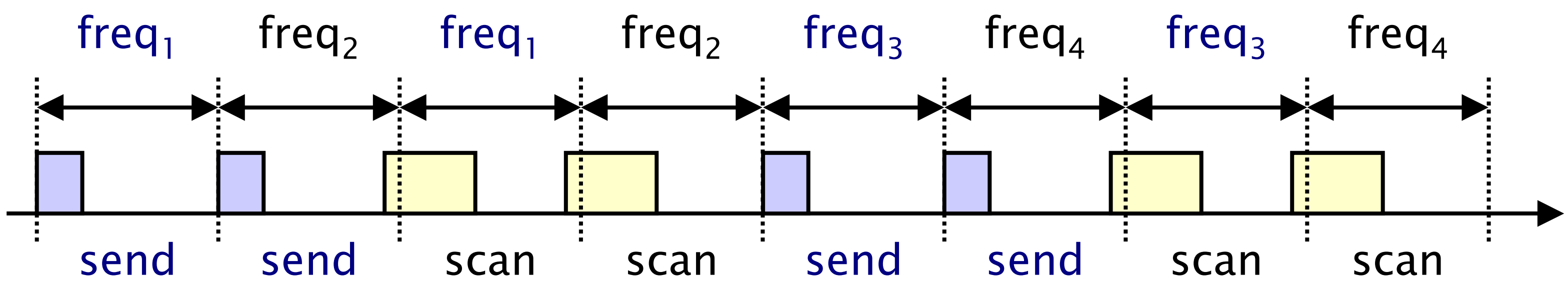
- Bluetooth: short-range low-power wireless protocol
  - widely available in phones, PDAs, laptops, ...
  - personal area networks (PANs)
  - open standard, specification freely available
- Uses frequency hopping scheme
  - to avoid interference (uses unregulated 2.4GHz band)
  - pseudo-random selection over 32 of 79 frequencies
- Network formation
  - piconets (1 master, up to 7 slaves)
  - self-configuring: devices discover themselves

Bluetooth device discovery

- States of a Bluetooth device:
  - standby: default operational state
  - inquiry: device discovery
    - master looks for devices, slaves listens for master
  - page: establish connection - synchronise clocks, etc.
  - connected: device ready to communicate in a piconet

- Device discovery
  - mandatory first step before any communication possible
  - “page” reuses information from “inquiry” so is much faster
  - power consumption much higher for “page”
  - performance crucial

Frequency hopping



- 28 bit free-running clock CLK, ticks every 312.5μs
- Master broadcasts inquiry packets on two consecutive frequencies, then listens on the same two (plus margin)
- Potential slaves want to be discovered, scan for messages
- Frequency sequence determined by formula, dependent on bits of clock CLK (k defined on next slide):

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

Master (sender) behaviour

- Broadcasts inquiry packets on two consecutive sequences, then listens on the same two
- Frequency hopping sequence determined by clock

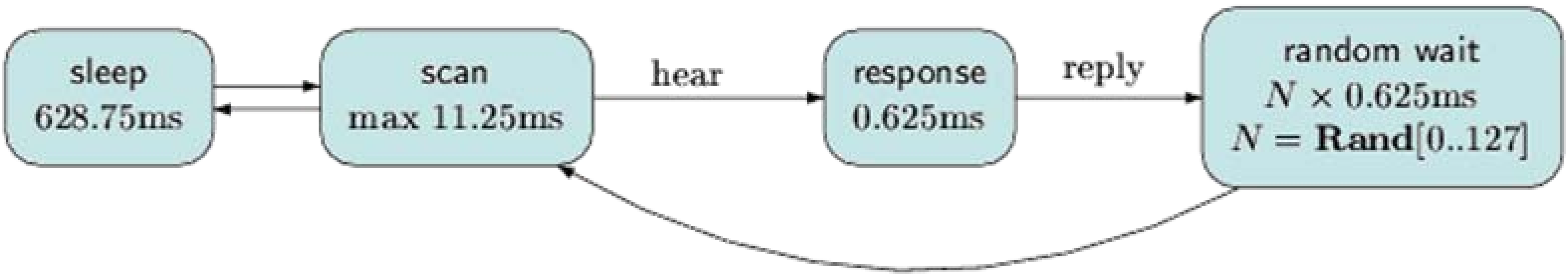
  freq = $[CLK_{16-12} + k + (CLK_{4-2,0} - CLK_{16-12}) \bmod 16] \bmod 32$

  - two trains (=lines) of 16 frequencies (determined by offset k)
  - each train repeated 128 times
  - swaps between trains every 2.56s

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

Slave (receiver) behaviour

- Listens (scans) on frequencies for inquiry packets
  - must listen on right frequency at right time
  - cycles through frequency sequence at much slower speed (every 1.28s)



- On hearing packet, pause, send reply and then wait for a random delay before listening for subsequent packets
  - avoid repeated collisions with other slaves

Bluetooth modelling

- Very complex interaction
  - genuine randomness, probabilistic modelling essential
  - devices make contact only if listen on the right frequency at the right time!
  - sleep/scan periods unbreakable, much longer than listening
  - cannot omit sub-activities, otherwise model is oversimplified

- Huge model, even for one sender and one receiver!
  - initial configurations dependent on 28 bit clock
  - cannot fix start state of receiver, clock value could be arbitrary

- But is a realistic future ubiquitous computing scenario!

Bluetooth – PRISM model

- Modelling in PRISM [DKNP06]
  - model one sender and one receiver
  - synchronous (clock speed defined by Bluetooth spec)
  - randomised behaviour – use DTMC
  - model at lowest-level (one clock-tick = one transition)
  - use real values for delays, etc, from Bluetooth spec

- Modelling challenges
  - complex interaction between sender/receiver
  - combination of short/long time-scales – cannot scale down
  - sender/receiver not initially synchronised, huge number of possible initial configurations (17,179,869,184)

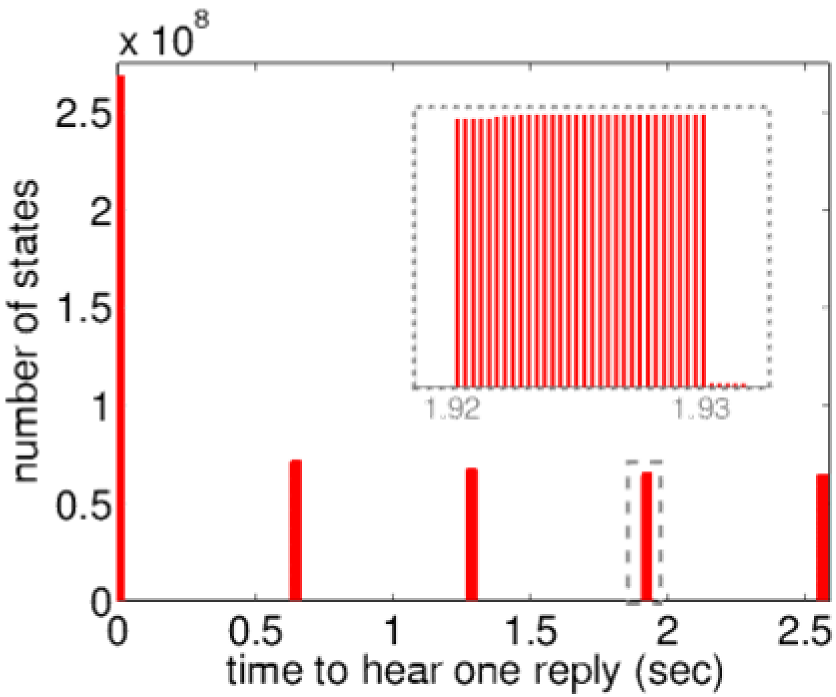
Bluetooth – Results

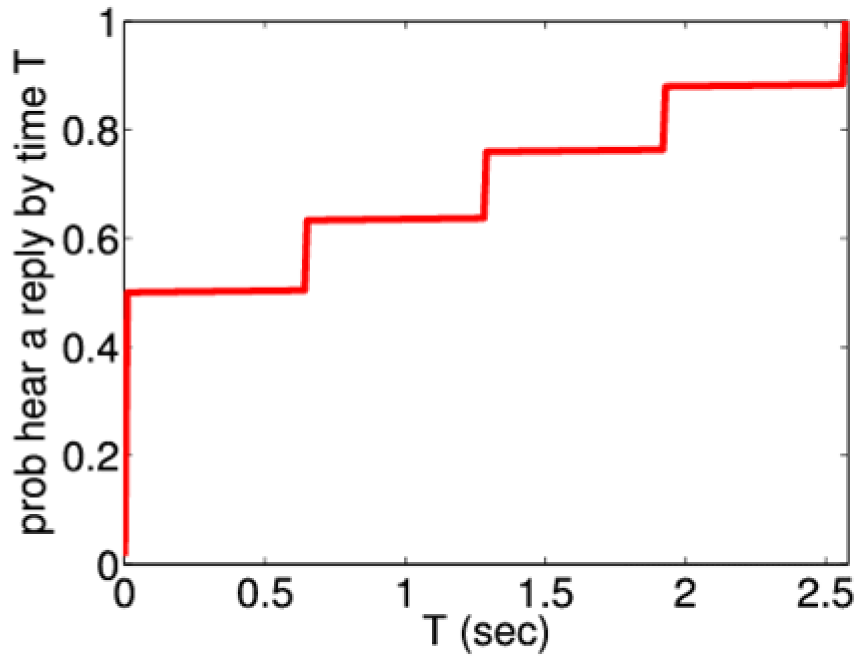
- Huge DTMC!
  - initially, model checking infeasible
  - partition into 32 scenarios, i.e. 32 separate DTMCs
  - on average, approx. 3.4×10^9 states, 536,870,912 initial
  - can be built/analysed with PRISM's MTBDD engine

- Property model checked:
  - $R_{=?}$ [F replies=K {"init"}{max}]
  - "worst-case (maximum) expected time to hear K replies, over all possible initial configurations"
  - also: how many initial states for each possible expected time
  - and: cumulative distribution function assuming equal probability for each initial state

Bluetooth – Time to hear 1 reply

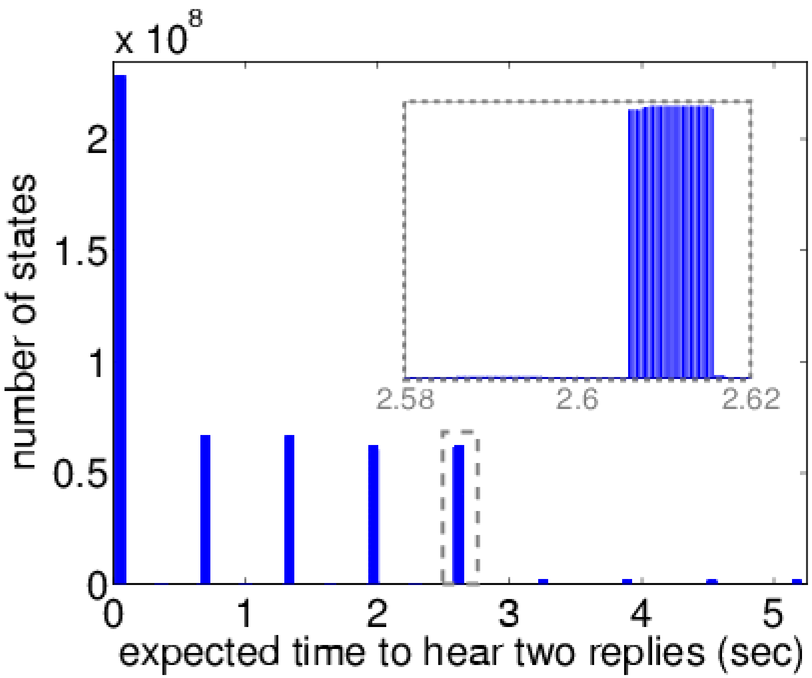
- Worst-case expected time = 2.5716s
  - in 921,600 possible initial states
- Best-case expected time = 635μs

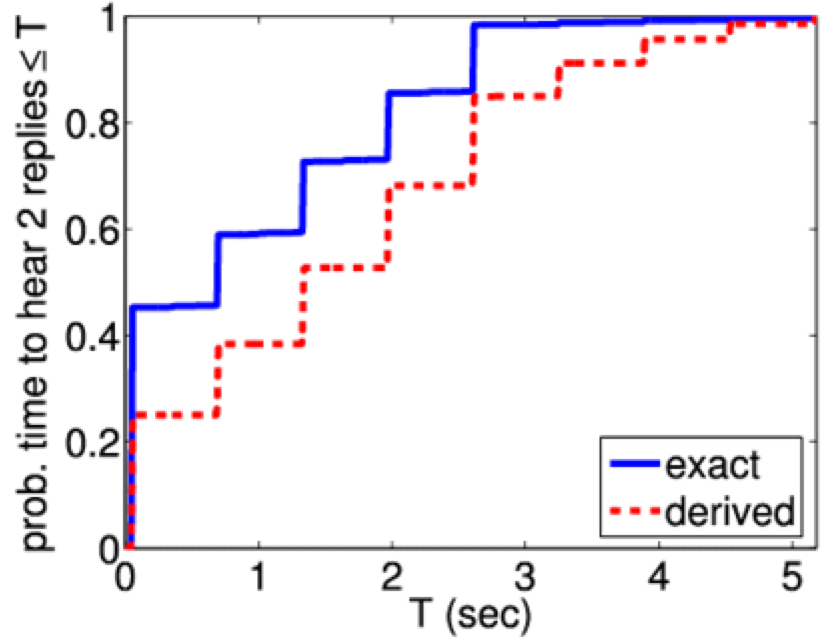




Bluetooth – Time to hear 2 replies

- Worst-case expected time = 5.177s
  - in 444 possible initial states
- Compare actual CDF with derived version which assumes times to reply to first/second messages are independent





Bluetooth – Results

- Other results (see [DKNP06])
  - compare versions 1.2 and 1.1 of Bluetooth, confirm 1.1 slower
  - power consumption analysis (using rewards)

- Conclusions
  - successful analysis of complex real-life model, actual parameters from standard
  - exhaustive analysis: best-/worst-case values
    - can pinpoint scenarios which give rise to them
    - not possible with simulation approaches
  - model still relatively simple
    - consider multiple receivers?
    - combine with simulation?

Contract signing

- Two parties want to agree on a contract
  - each will sign if the other will sign, but do not trust each other
  - there may be a trusted third party (judge)
  - but it should only be used if something goes wrong
- In real life: contract signing with pen and paper
  - sit down and write signatures simultaneously
- On the Internet…
  - how to exchange commitments on an asynchronous network?
  - "partial secret exchange protocol" [EGL85]

Contract signing – EGL protocol

- Partial secret exchange protocol for 2 parties (A and B)

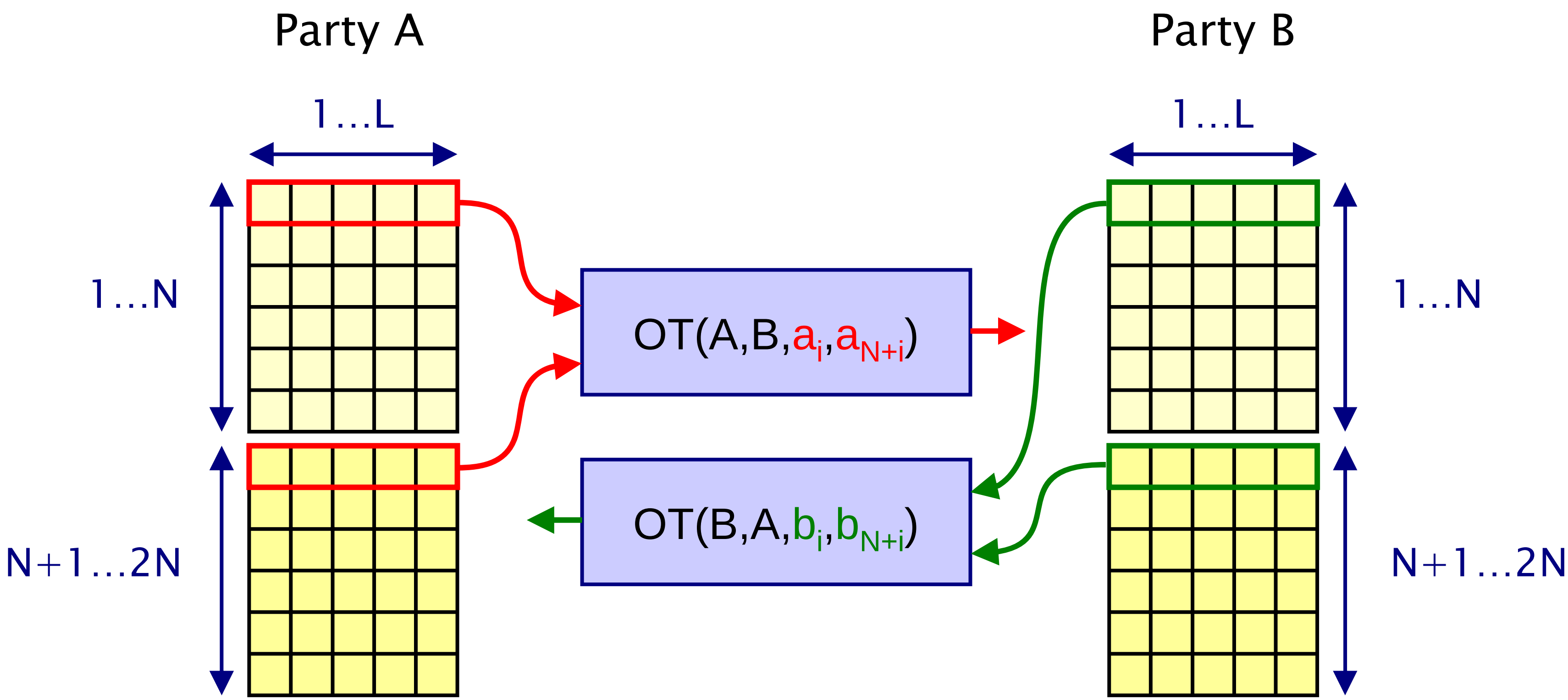
- A (B) holds 2N secrets $a_1,\ldots,a_{2N}$ $(b_1,\ldots,b_{2N})$
  - a secret is a binary string of length L
  - secrets partitioned into pairs: e.g. $\{ (a_i, a_{N+i}) \mid i=1,\ldots,N \}$
  - A (B) committed if B (A) knows one of A's (B's) pairs

- Uses "1-out-of-2 oblivious transfer protocol" OT(S,R,x,y)
  - S sends x and y to R
  - R receives x with probability ½ otherwise receives y
  - S does not know which one R receives
  - if S cheats then R can detect this with probability ½

Contract signing – EGL protocol

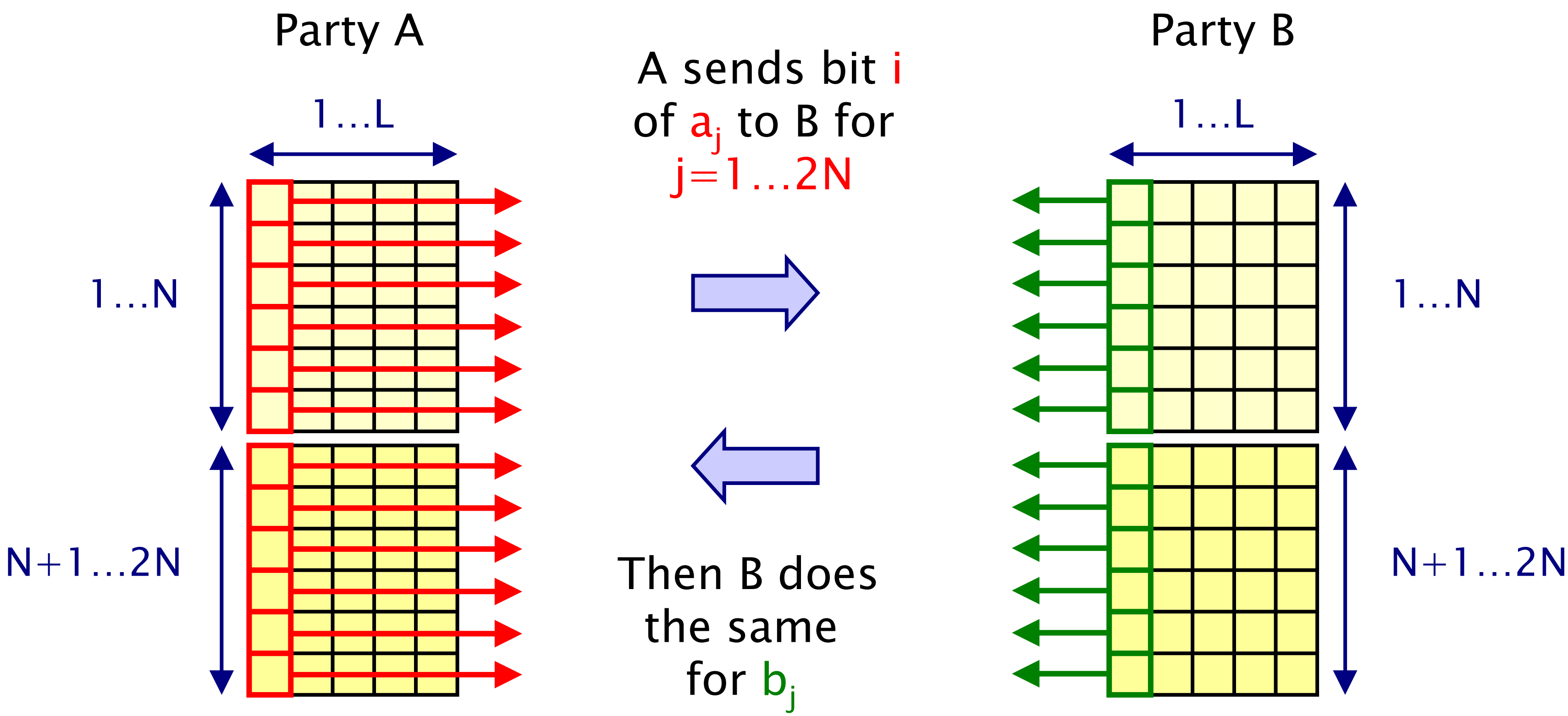
(step 1)
for (i=1,…,N)
    OT(A, B, a_i, a_{N+i})
    OT(B, A, b_i, b_{N+i})
(step 2)
    for (i=1,…,L)   (where L is the bit length of the secrets)
        for (j=1,…,2N)
            A transmits bit i of secret a_j to B
        for (j=1,…,2N)
            B transmits bit i of secret b_j to A

EGL protocol – Step 1



(repeat for i=1…N)

EGL protocol – Step 2



(repeat for i=1…L)

Contract signing – Results

- Modelled in PRISM as a DTMC (no concurrency) [NS06]
- Discovered a weakness in the protocol
  - party B can act maliciously by quitting the protocol early
  - this behaviour not considered in the original analysis
- PRISM analysis shows
  - if B stops participating in the protocol as soon as he/she has obtained one of A pairs, then, with probability 1, at this point:
    - B possesses a pair of A's secrets
    - A does not have complete knowledge of any pair of B's secrets
  - protocol is not fair under this attack:
  - B has a distinct advantage over A

Contract signing – Results

- The protocol is unfair because in step 2:
  – A sends a bit for each of its secret before B does

- Can we make this protocol fair by changing the message sequence scheme?

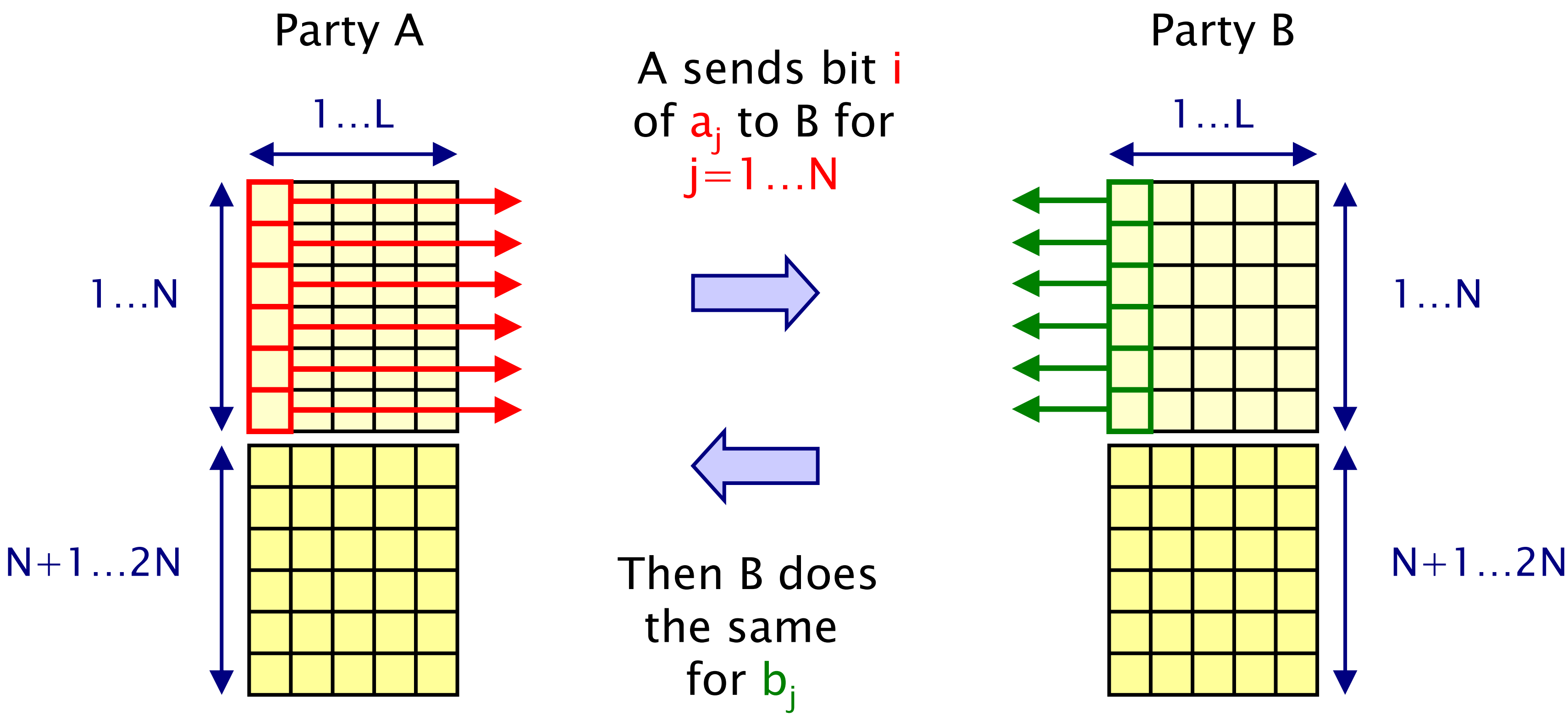
- Since the protocol is asynchronous the best we can hope for is
  – B (or A) has this advantage with probability ½

- We consider 3 possible alternative message sequence schemes…

Contract signing – EGL2

(step 1)
…
(step 2)
for (i=1,…,L)
    for (j=1,…,N) A transmits bit i of secret a_j to B
    for (j=1,…,N) B transmits bit i of secret b_j to A
    for (j=N+1,…,2N) A transmits bit i of secret a_j to B
    for (j=N+1,…,2N) B transmits bit i of secret b_j to A

Modified step 2 for EGL2



(after j=1…N, send j=N+1…2N)
(then repeat for i=1…L)

Contract signing – EGL3

(step 1)

…

(step 2)

for (i=1,…,L) for (j=1,…,N)

  A transmits bit i of secret a_j to B

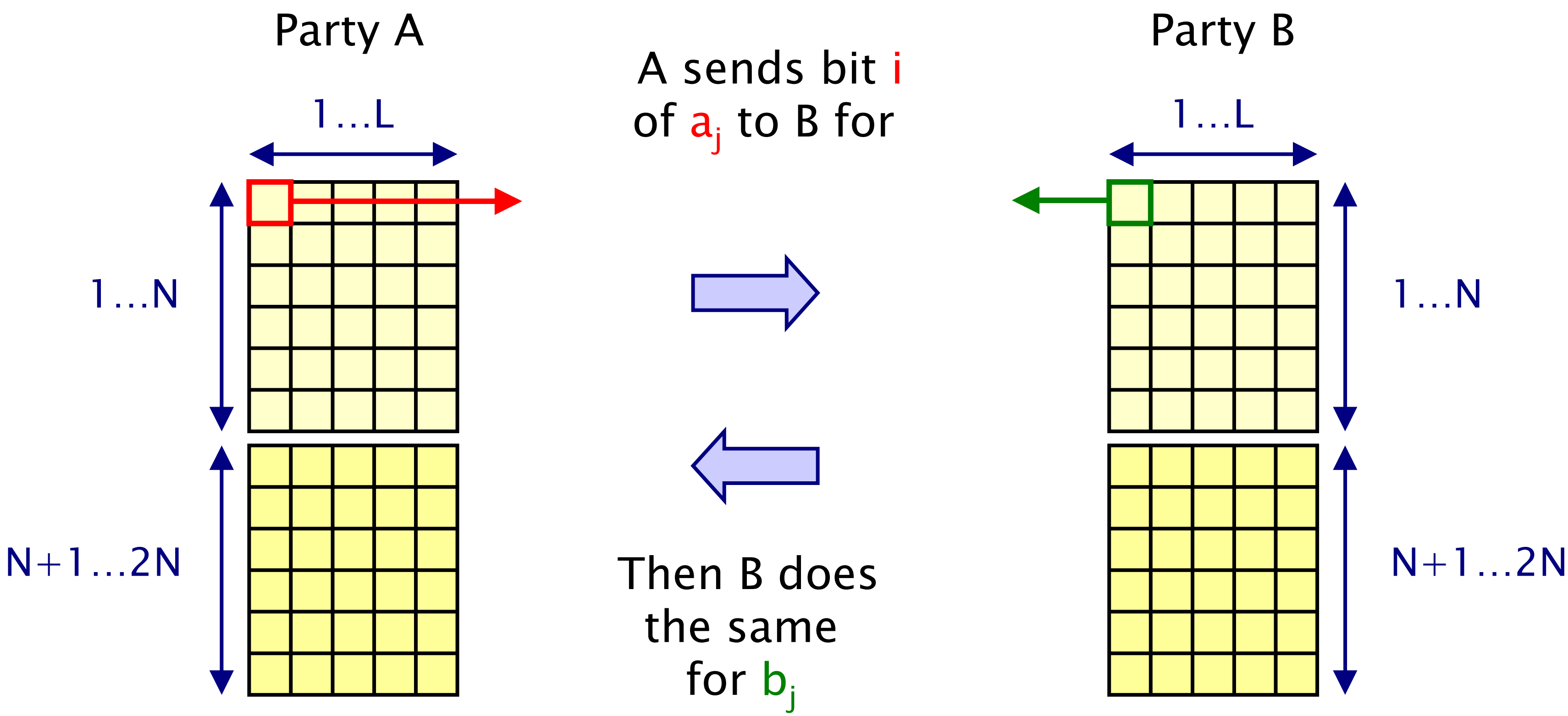
  B transmits bit i of secret b_j to A

for (i=1,…,L) for (j=N+1,…,2N)

  A transmits bit i of secret a_j to B

  B transmits bit i of secret b_j to A

Modified step 2 for EGL3

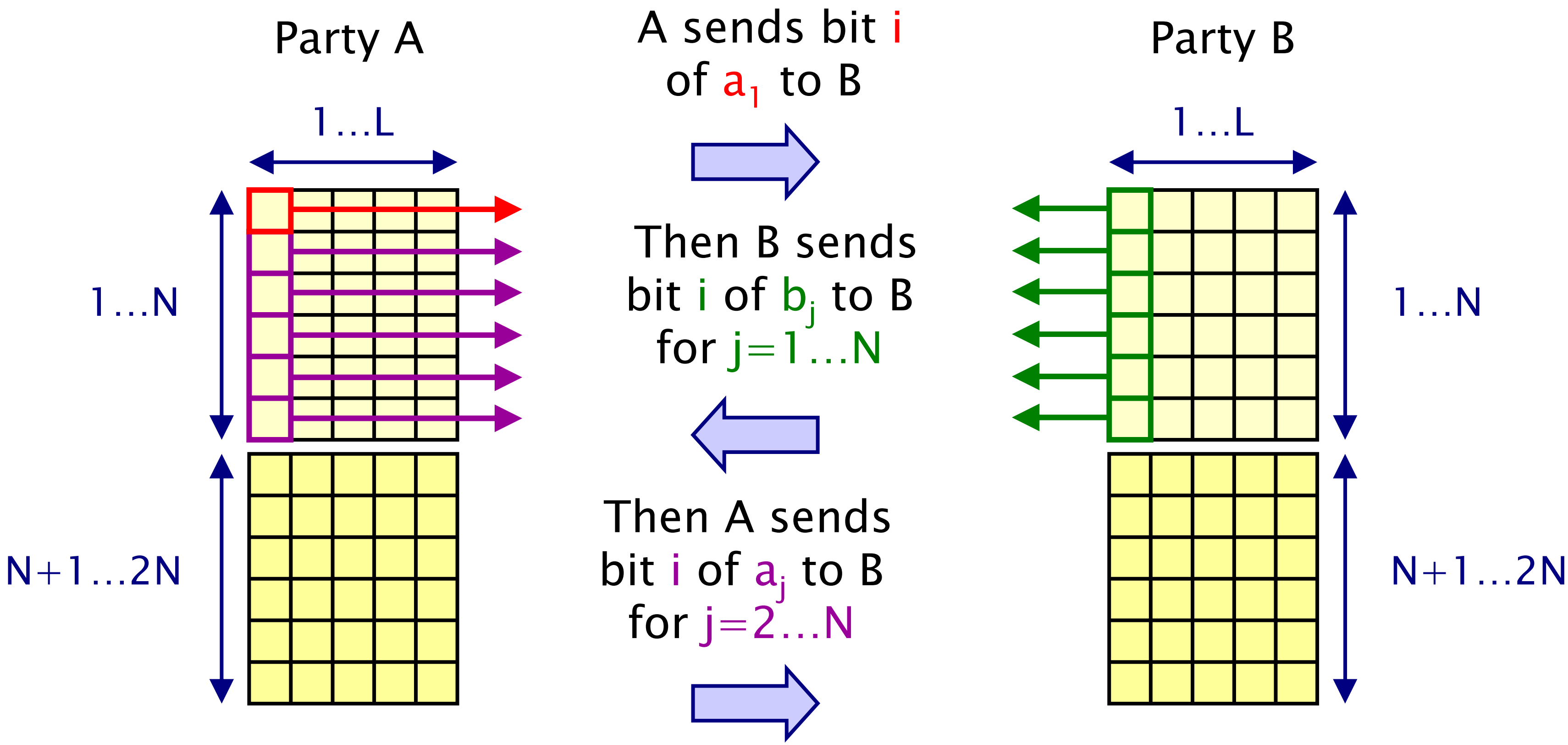


(repeat for $j=1\ldots N$ and for $i=1\ldots L$)
(then send $j=N+1\ldots 2N$ for $i=1\ldots L$)

Contract signing – EGL4

(step 1)
…
(step 2)
for (i=1,…,L)
  A transmits bit i of secret a_1 to B
    for (j=1,…,N) B transmits bit i of secret b_j to A
    for (j=2,…,N) A transmits bit i of secret a_j to B
for (i=1,…,L)
  A transmits bit i of secret a_{N+1} to B
    for (j=N+1,…,2N) B transmits bit i of secret b_j to A
    for (j=N+2,…,2N) A transmits bit i of secret a_j to B

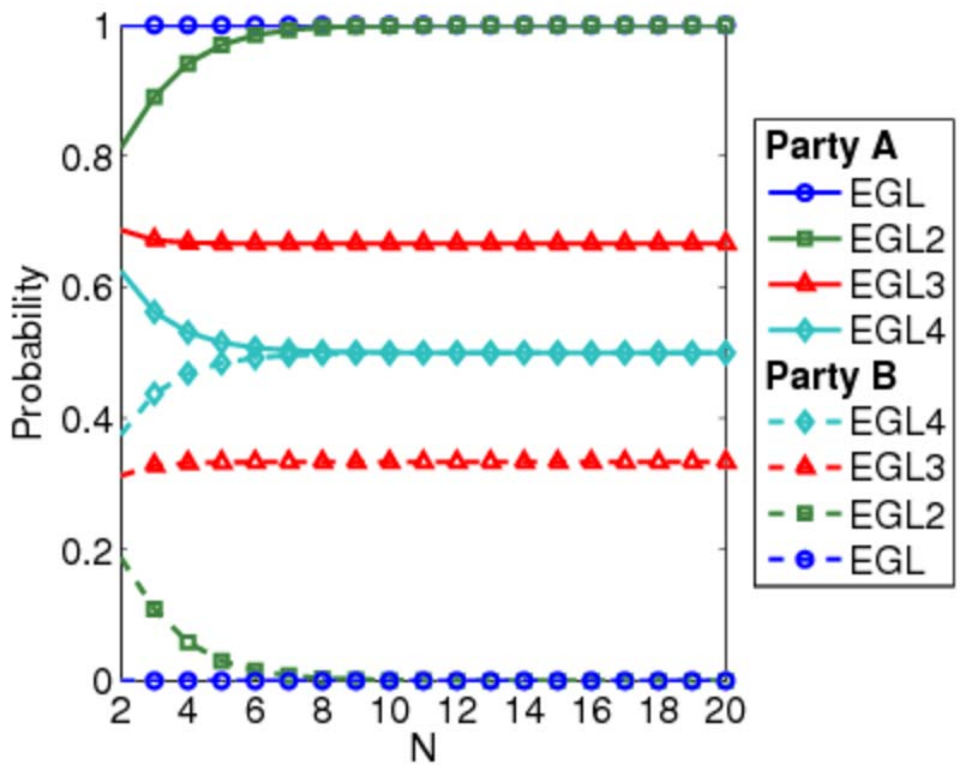
Modified step 2 for EGL4



(repeat for i=1…L)
(then send j=N+1…2N in same fashion)

Contract signing - Results

- The chance that the protocol is unfair
  - probability that one party gains knowledge first
  - $P_{=?}[F\ know_B \wedge \neg know_A]$ and $P_{=?}[F\ know_A \wedge \neg know_B]$



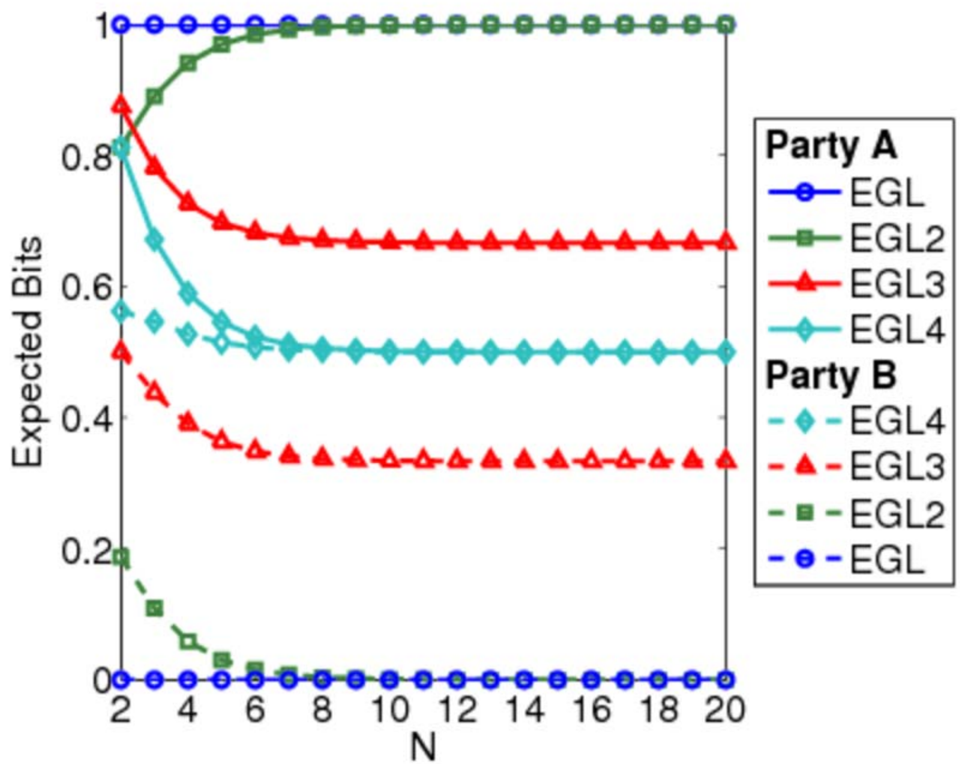
Contract signing – Results

- How unfair the protocol is to each party
  - expected number of bits that a party needs to know a pair once the other party knows a pair
  - need to modify the model and define a reward structure
  - dependent on which party we are considering

- Expected number of bits that A needs to know a pair once B knows a pair
  - add a transition to a new state labelled by “done” as soon as B knows a pair
  - assign a reward equal to the number of bits that A requires to know a pair to this transition
  - check the formula $R_{=?}[F \text{ done}]$

Contract signing – Results

- How unfair the protocol is to each party
  - expected number of bits that a party needs to know a pair once the other party knows a pair



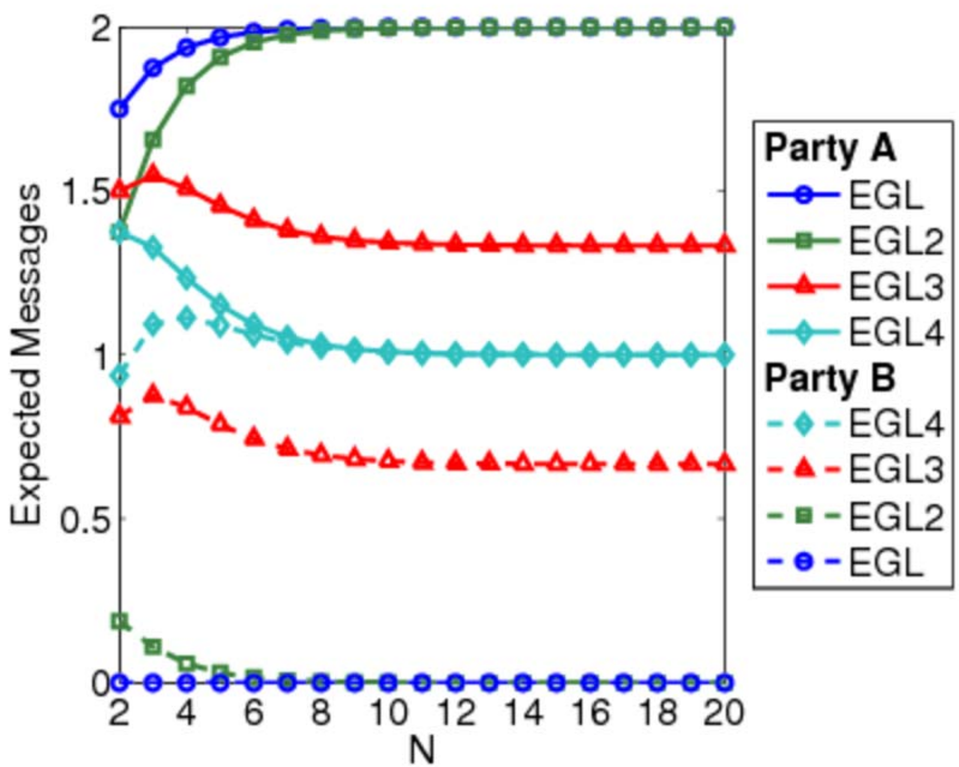
Contract signing – Results

- The influence that each party has on the fairness
  - once a party knows a pair, the expected number of messages from this party required before the other party knows a pair
  - measures the influence as a corrupted party can delay its messages
  - need to define a reward structure
  - dependent on which party we are considering

- Once B knows a pair, the expected number of messages from B required before A knows a pair
  - assign reward of 1 to transitions which correspond to B sending a message to A from a state where B knows a pair
  - check the formula $R_{=?}[F\ know_A]$

Contract signing – Results

- The influence the each party has on the fairness
  - once a party knows a pair, the expected number of messages from this party required before the other party knows a pair

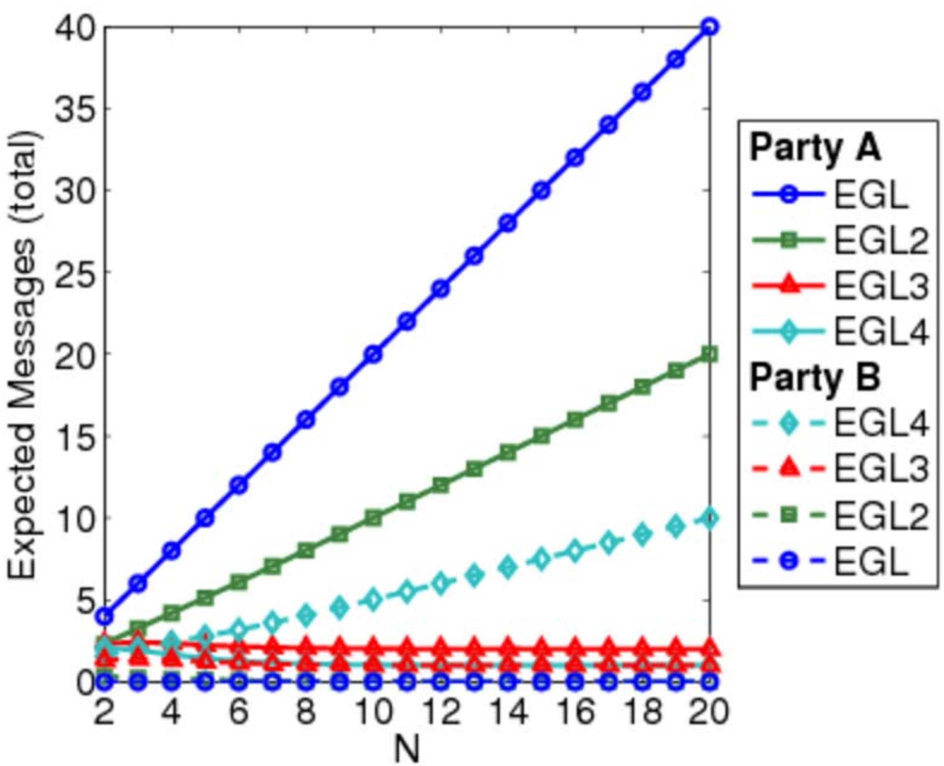


Contract signing – Results

- The duration of unfairness of the protocol
  - once a party knows a pair, the expected total number of messages that need to be sent (by either party) before the other knows a pair
  - need to define a reward structure
  - dependent on which party we are considering
- Once B knows a pair, the expected total number of messages that need to be sent before A knows a pair
  - assign reward of 1 to transitions which correspond to either party sending a message from a state where B knows a pair
  - check the formula $R_{=?}$[F $know_A$]

Contract signing – Results

- The duration of unfairness of the protocol
  - once a party knows a pair, the expected total number of messages that need to be sent before the other knows a pair

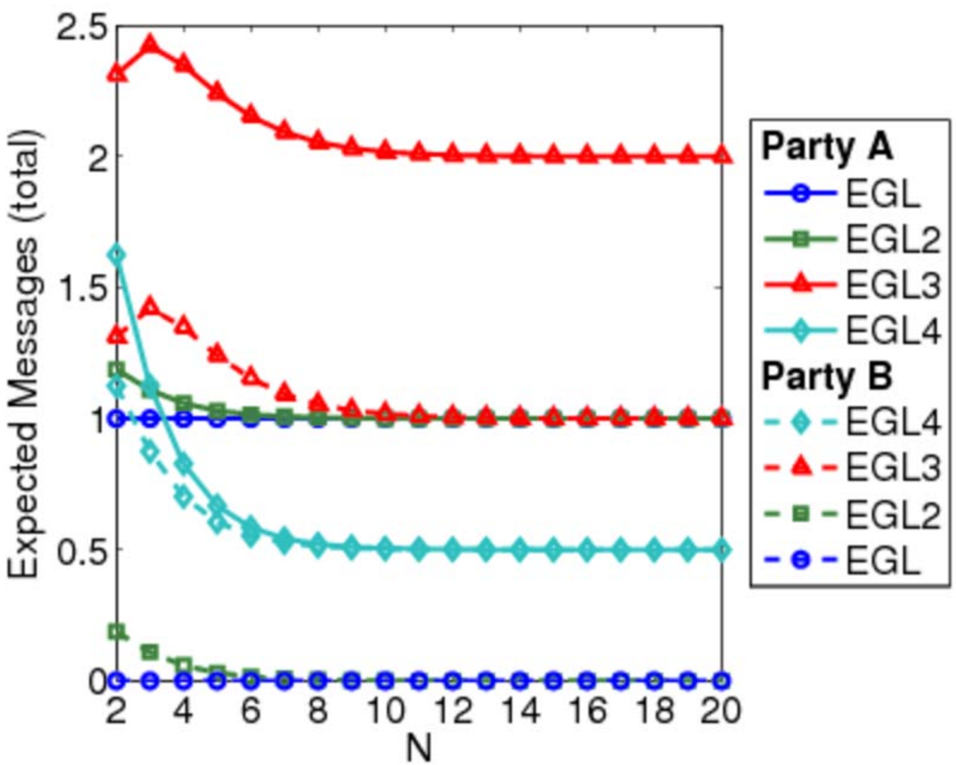


Contract signing – Results

- Results show EGL4 is the 'fairest' protocol
- Except for duration of fairness measure…
- Expected messages that need to be sent for a party to know a pair once the other party knows a pair
  - this value is larger for B than for A
  - in fact, as n increases, this measure increases for B and decreases for A
- Solution
  - if a party sends a sequence of bits in a row (without the other party sending messages in between), require that the party send these bits as as a single message

Contract signing – Results

- The duration of unfairness of the protocol
  - once a party knows a pair, the expected total number of messages that need to be sent before the other knows a pair



Summing up…

- What have we achieved?
- For Bluetooth device discovery,
  - for the first time, obtained exact worst case expected response time to 1 message, and likewise for 2 messages
  - can pinpoint the cause, impossible with simulation
  - BTW, it is 2.5 seconds!
  - no wonder Bluetooth gets criticised for being slow…
- For contract signing
  - identified an assumption missed by the authors
  - proposed a fix

Further information

- More on the Bluetooth case study
  - see [DKNP06]
- More on contract signing
  - see [NS06]

- More on similar protocols
  - Crowds anonymity [Shm04]
  - probabilistic anonymity [BP05]
  - PIN cracking [Ste06]

- More information, see the PRISM web page

  www.prismmodelchecker.org