Analysing mobile ad hoc networks via probabilistic model checking

Marta Kwiatkowska
School of Computer Science

www.cs.bham.ac.uk/~mzk
www.cs.bham.ac.uk/~dxp/prism

MSR Redmond, April 2005
Overview

- **Mobile ad hoc network protocols**
  - Probability - why needed, challenges
  - Verification techniques and tools

- **Probabilistic model checking**
  - The models
  - Specification languages
  - What does it involve?
  - The PRISM model checker

- **Case studies**
  - IPv4 Zeroconf dynamic configuration protocol
  - Bluetooth device discovery

- **Challenges for future**
Ubiquitous computing: the trends...

- Devices, ever smaller
  - Laptops, phones, PDAs, ...
  - Sensors, motes, ...
- Networking, wireless, wired & global
  - Mobile ad hoc
  - Wireless everywhere
  - Internet everywhere
  - Global connectivity
- Systems/software
  - Decentralised
  - Self-organising
  - Self-configuring
  - Autonomous
  - Adaptive
  - Context-aware
Ubiquitous computing: users expect...

• ...assurance of
  - safety
  - correctness
  - performance
  - reliability

• For example:
  - Is my e-savings account secure?
  - Can someone bluesnarf from my phone?
  - How fast is the communication from my PDA to printer?
  - Is my mobile phone energy efficient?
  - Is the protocol reliable?
  - Can the laptop recover from faults with no effort on my part?
Probability helps

- In distributed (de-centralised) co-ordination algorithms
  - As a symmetry breaker
    - "leader election is eventually resolved with probability 1"
  - In gossip-based routing and multicasting
    - "the message will be delivered to all nodes with high probability"

- When modelling uncertainty in the environment
  - To quantify failures, express soft deadlines, QoS
    - "probability of frame being delivered within 5ms is at least 0.91"
  - To quantify environmental factors in decision support
    - "expected cost of reaching the goal is 100"

- When analysing system performance
  - To quantify arrivals, service, etc. characteristics
    - "in the long run, mean waiting time in a lift queue is 30 sec"
Real-world protocol examples

- Protocols featuring randomisation
  - Randomised back-off schemes
    - IEEE 802.11 (WiFi) Wireless LAN MAC protocol
  - Random choice of waiting time
    - Bluetooth, device discovery phase
  - Random choice of routes to destination
    - Crowds, anonymity protocol for internet routing
  - Random choice of a timing delay
    - Root contention in IEEE 1394 FireWire
  - Random choice over a set of possible addresses
    - IPv4 dynamic configuration (link-local addressing)
  - and more

- Continuous probability distribution needed to model network traffic, node mobility, random delays...
Verification via model checking... or falsification?

The model

send → ◊deliver

Temporal logic specification

Model Checker

Error trace

Line 5: ...
Line 21: ...
Line 15: ...
... Line 27: ...
Line 45: ...
Probabilistic model checking...

in a nutshell

send → P_{0.9}(\diamond deliver)

Probabilistic model

Probabilistic model checker

State 5: 0.6789
State 6: 0.9789
State 7: 1.0
...
State 12: 0
State 13: 0.1245

-or-

The probability
Probability elsewhere

• **In performance modelling**
  - Pioneered by Erlang, in telecommunications, ca 1910
  - Models: typically continuous time Markov chains
  - Emphasis on steady-state and transient probabilities

• **In stochastic planning**
  - Cf Bellman equations, ca 1950s
  - Models: Markov decision processes
  - Emphasis on finding optimum policies

• **Our focus, probabilistic model checking**
  - Distinctive, on automated verification for probabilistic systems
  - Temporal logic specifications, automata-theoretic techniques
  - Shared models
  - Exchanging techniques with the other two areas
Probabilistic models: discrete time

- **Labelled transition systems**
  - Discrete time steps
  - Labelling with atomic propositions

- **Probabilistic transitions**
  - Move to state with given probability
  - Represented as discrete probability distribution

- **Model types**
  - Discrete time Markov chains (DTMCs): probabilistic choice only
  - Markov decision processes (MDPs): probabilistic choice and nondeterminism

\[
\sum_i p_i = 1
\]
Discrete-Time Markov Chains (DTMCs)

- **Features:**
  - Only probabilistic choice in each state

- **Formally, \((S, s_0, P, L)\):**
  - \(S\) finite set of states
  - \(s_0\) initial state
  - \(P: S \times S \to [0,1]\) probability matrix, s.t. \(\sum_{s'} P(s, s') = 1\), all \(s\)
  - \(L: S \to 2^{AP}\) atomic propositions

- **Unfold into infinite paths** \(s_0 s_1 s_2 s_3 s_4 \ldots\) s.t. \(P(s_i, s_{i+1}) > 0\), all \(i\)

- **Probability for finite paths**, multiply along path
  e.g. \(s_0 s_1 s_1 s_2\) is \(1 \cdot 0.01 \cdot 0.97 = 0.0097\)
### Probability space

- **Intuitively:**
  - Sample space = infinite paths \( \text{Path}_s \) from \( s \)
  - Event = set of paths
  - Basic event = cone

- **Formally, \((\text{Path}_s, \Omega, \Pr)\):**
  - For finite path \( \omega = ss_1s_2...s_k \), define probability
    
    \[
    P(\omega) = \begin{cases} 
      1 & \text{if } \omega \text{ has length one} \\
      P(s,s_1) \land ... \land P(s_{n-1},s_n) & \text{otherwise}
    \end{cases}
    \]

  - Take \( \Omega \) least \( \sigma \)-algebra containing cones
    
    \[ C(\omega) = \{ \pi \text{ 2 Path}_s \mid \omega \text{ is prefix of } \pi \} \]
  - Define \( \Pr(C(\omega)) = P(\omega) \), all \( \omega \)
  - \( \Pr \) extends uniquely to measure on \( \text{Path}_s \)
Markov Decision Processes (MDPs)

- Features:
  - Nondeterministic choice
  - Parallel composition of DTMCs

- Formally, \((S, s_0, \text{Steps}, L)\):
  - \(S\) finite set of states
  - \(s_0\) initial state
  - \(\text{Steps}\) maps states \(s\) to sets of probability distributions \(\mu\) over \(S\)
  - \(L: S \rightarrow 2^{AP}\) atomic propositions

- Unfold into infinite paths \(s_0\mu_0s_1\mu_1s_2\mu_2s_3\ldots\) s.t. \(\mu_i(s_i, s_{i+1}) > 0\), all \(i\)

- Probability space induced on \(\text{Path}_s\) by adversary (policy) \(A\) mapping finite path \(s_0\mu_0s_1\mu_1\ldots s_n\) to a distribution from state \(s_n\)
The logic PCTL: syntax

- **Probabilistic Computation Tree Logic** [HJ94,BdA95,BK98]
  - For DTMCs/MDPs
  - New probabilistic operator, e.g. `send \to P_{>0.9}(\Diamond \text{deliver})`
    “whenever a message is sent, the probability that it is eventually delivered is at least 0.9”

- The syntax of state and path formulas of PCTL is:

  \[
  \phi ::= \text{true} \mid a \mid \phi \land \phi \mid \phi \lor \phi \mid P_{p}(\alpha)
  \]
  \[
  \alpha ::= X \phi \mid \phi U \phi
  \]
  
  where \( p \in [0,1] \) is a probability bound and \( \rightarrow \) \{ <, >, ... \}

- Subsumes the qualitative variants [Var85,CY95] \( P_{=1}(\alpha), P_{>0}(\alpha) \)

- Extension with \textit{cost/rewards} and \textit{expectation} operator \( E_{c}(\phi) \)
The logic PCTL: semantics

- Semantics is parameterised by a class of adversaries $\text{Adv}$
  - “under any scheduling, the probability bound is true at state $s$”
  - reasoning about worst-case/best-case scenario

- The probabilistic operator is a quantitative analogue of 8, 9

\[
S \overset{2_{\text{Adv}}}{\mathbb{P}} \overset{p(\alpha)}{\Rightarrow} \quad \text{Pr}^A \{ \pi 2 \text{Path}^A_{s} j \pi 2_{\text{Adv}} \alpha \} \Rightarrow p
\]

for all $A \in \text{Adv}$
PCTL semantics: summary

- Semantics of state formulas:
  \[ s^{\text{Adv}} a \], \[ s^{\text{Adv}} :\phi \], \[ s^{\text{Adv}} \phi_1 \land \phi_2 \]
  \[ a \ 2 \ L(s) \], \[ s^{\text{Adv}} \phi \], \[ s^{\text{Adv}} \phi_1 \land s^{\text{Adv}} \phi_2 \]

- Semantics of path formulas:
  \[ \pi^{\text{Adv}} X \phi \], \[ \pi^{\text{Adv}} \phi_1 \lor \phi_2 \]
  \[ \pi = s_0 \ldots \land s_1^{\text{Adv}} \phi \], \[ \pi = s_0 \ldots \land 9 \ k \ s.t. \]
  \[ s_k^{\text{Adv}} \phi_2 \land 8 \ j < k \ . \ s_j^{\text{Adv}} \phi_1 \]

- The probabilistic operator:
  \[ s^{\text{Adv}} P >_p (\alpha) \]
  \[ \Pr^{A} \{ \pi 2 \ \text{Path}^{A}_s j \pi^{\text{Adv}} \alpha \} >_p \]
  for all \( A 2 \ \text{Adv} \)
The logic PCTL: model checking

- By induction on structure of formula, as for CTL

- For the probabilistic operator and Until, solve
  - recursive linear equation for DTMCs
  - linear optimisation problem (form of Bellman equation) for MDPs
  - typically iterative solution methods

- Need to combine
  - conventional graph traversal
  - numerical linear algebra and linear optimisation (value iteration)

- Qualitative properties (probability 1, 0) proceed by graph traversal [Var85,dAKNP97]
PCTL model checking for DTMCs

- By induction on structure of formula
- For the probabilistic operator
  - $\text{Sat}( P\cdot P(X\ \phi) )$,  \( \{ s \in S | \sum_{s'} P(s,s') \cdot \phi \} \)
  - $\text{Sat}( P\cdot (\phi_1 \cup \phi_2) )$,  \( \{ s \in S | x_s \cdot P \} \)

where $x_s$, $s \in S$, are obtained from the recursive linear equation

$$x_s = \begin{cases} 
0 & \text{if } s \in S^{\text{no}} \\
1 & \text{if } s \in S^{\text{yes}} \\
\sum_{s' \in S} P(s,s') \cdot x_{s'} & \text{if } s \in S^n (S^{\text{no}} \cup S^{\text{yes}}) 
\end{cases}$$

and

$S^{\text{yes}}$ - states that satisfy $\phi_1 \cup \phi_2$ with probability exactly 1
$S^{\text{no}}$ - states that satisfy $\phi_1 \cup \phi_2$ with probability exactly 0
PCTL model checking for DTMCs

• For the remaining formulas standard:

\[
\begin{align*}
\text{Sat}(a) &= L(a) \\
\text{Sat}(\phi) &= S \setminus \text{Sat}(\phi) \\
\text{Sat}(\phi_1 \land \phi_2) &= \text{Sat}(\phi_1) \setminus \text{Sat}(\phi_2)
\end{align*}
\]

• Syes, Sno can be precomputed by graph traversal [Var85] (or BDD fixed point computation)

• Need to combine
  - Conventional graph-theoretic traversal
  - Numerical linear algebra
PCTL model checking for MDPs

- $S_{yes}, S_{no}$ can also be precomputed by graph traversal (BDD fixed point) [dAKNP97]

- The linear equation generalises to linear optimisation problems solvable iteratively, e.g.

$$\text{Sat}( P_{s} (\phi_1 \cup \phi_2) ) , \{s 2 S \mid x_s , p\}$$

$$x_s = \begin{cases} 
0 & \text{if } s 2 S_{no} \\
1 & \text{if } s 2 S_{yes} \\
\min_{\mu \text{ 2 Steps}(s)} \sum_{s' 2 S} \mu(s') \not\in x_{s'} & \text{if } s 2 S_{n}(S_{no} [ S_{yes} )
\end{cases}$$

- Need to combine
  - Conventional graph-theoretic traversal
  - Linear optimisation (simplified value iteration)
Probabilistic models: continuous

- **Assumptions on time and probability**
  - Continuous passage of time
  - Continuous randomly distributed delays
  - Continuous space

- **Model types**
  - Continuous time Markov chains (CTMCs): exponentially distributed delays, discrete space, no nondeterminism
  - Probabilistic Timed Automata (PTAs): dense time, (usually) discrete probability, admit nondeterminism
  - (not considered) Labelled Markov Processes (LMPs): continuous space/time, no nondeterminism

\[ S_0 \int_0^1 f(x)dx = 1 \]
Probabilistic model checking in practice

- **Model construction:** probability/rate matrices
  - **Enumerative**
    - Manipulation of *individual* states
    - Size of state space main limitation
  - **Symbolic**
    - Manipulation of *sets* of states
    - Compact representation possible in case of regularity

- **Temporal logic** model checking: currently limited to
  - discrete probability/space models
  - CTMCs
  - Simulation admits more general distributions

- **Probabilistic Symbolic Model Checker** PRISM
The PRISM project

• History
  - First public release September 2001, ~7 years development
  - **Connection with other software tools:** KRONOS, PEPA, CSP/FDR2, APMC, YMER
  - **Forthcoming:** BioCHAM, CADP, Probmela

• Staff
  - **Core:** Kwiatkowska, RFs (EPSRC): Parker, Norman, Zhang
  - PhD students: Honore (mobility), Tymchyshyn (biology)
  - **Key collaborators:** Younes (CMU), Shmatikov (SRI/Texas), Segala (Verona), Katoen (Twente/Aachen), Baier (Bonn), Shukla (VT), Gilmore (Edinburgh), Goldsmith (Formal Sys), and more

• Users
  - 2000 downloads, Unix/Linux, Windows and Apple Mac
  - 30+ case studies, 70 papers featuring PRISM
  - Taught at Stanford, Austin Texas, KTH, Rome
The PRISM tool: overview

- **Functionality**
  - Direct support for models: DTMCs, MDPs and CTMCs
  - Extension with costs/rewards, expectation operator
  - PTAs with digital clocks by manual translation
  - Connection from KRONOS to PRISM for PTAs
  - Experimental implementation using DBMs/DDDs for PTAs

- **Input languages**
  - System description
    - Probabilistic extension of reactive modules [Alur and Henzinger]
  - Probabilistic temporal logics: PCTL and CSL

- **Implementation**
  - Symbolic model construction (MTBDDs), uses CUDD [Somenzi]
  - Three numerical computation engines
  - Written in Java and C++
The PRISM tool: implementation

• Numerical engines
  - **Symbolic**, MTBDD based
    • Fast construction, reachability analysis
    • Very large models if regularity
  - **Enumerative**, sparse-matrix based
    • Generally fast numerical computation
    • Model size up to millions
  - **Hybrid**
    • Speed comparable to sparse matrices for numerical calculations
    • Limited by size of vector

• Experimental results
  - Several large scale examples: $10^{10} - 10^{30}$ states
  - No engine wins overall
  - See [www.cs.bham.ac.uk/~dxp/prism](http://www.cs.bham.ac.uk/~dxp/prism)
PRISM real-world case studies

- **MDPs/DTMCs**
  - Self-stabilising algorithms (based on Hermann and others)
  - Bluetooth device discovery [ISOLA’04]
  - Crowds anonymity protocol (by Shmatikov) [CSFW’02, JSC 2003]
  - Randomised consensus [CAV’01, FORTE’02]
  - Contract signing protocols (by Norman & Shmatikov) [FASEC’02]
  - NAND multiplexing for nano (with Shukla) [VLSI’04, TCAD 2005]

- **CTMCs**
  - Molecular reactions (based on Regev & Shapiro)
  - Eukaryotic cell cycle control (based on Lecca & Priami)
  - Dependability of embedded controller [INCOM’04]
  - Dynamic power management [HLDVT’02, FAC 2005]

- **PTAs**
  - IPv4 ZeroConf dynamic configuration [FORMATS’03]
  - Root contention in IEEE 1394 FireWire [FAC 2003, STTT 2004]
  - IEEE 802.11 (WiFi) Wireless LAN MAC [PROBMIV’02, CAV’05]
PRISM technicalities

- Augment states and transitions with real-valued rewards
  - Instantaneous rewards, e.g. “concentration of reactant”
  - Cumulative rewards, state- and transition-based, e.g. “power consumed”, “messages lost”

- Support for “experiments”
  - e.g. P=\? [true U<=T error] for N=1..5,T=1..100

- GUI implementation
  - integrated editor for PRISM language
  - automatic graph plotting

- (Ongoing) Simulator and sampling-based model checking
  - allows to “execute” the model step-by-step or randomly
  - avoids state-space explosion, trading off accuracy
Screenshot: Text editor

```
module coin

// constants
const int HEADS = 1;
const int TAILS = 2;

// a single module
module coin

    // variable
    x : [0..3] init 0;

    // guarded commands
    (x = 0) -> 0.5 : (x' = HEADS) + 0.5 : (x' = TAILS);
    (x > 0) -> 1 : (x' = x);

endmodule
```
Screenshot: Graphs
Ongoing developments

- Graphical modelling language
- Simulator, sampling methods
- Parallel engine
- Grid engine
Case Study: Self-stabilization

• Self-stabilizing protocol for a network of processes
  - starts from possibly illegal start state
  - returns to a legal (stable) state
    • without any outside intervention
    • within some finite number of steps

• Network: synchronous or asynchronous ring of \( N \) processes
  - Illegal states: more than one process is privileged (has a token)
  - Stable states: exactly one process is privileged (has a token)
  - Properties
    • From any state, a stable state is reached with probability 1
    • Expected time to reach a stable state
    • Interested in worst-case time to reach stable state (unproven conjecture about Hermann’s ring of McIver & Morgan)
Herman's self-stabilising protocol

- Synchronous ring of \( N \) (\( N \) odd) processes (\textbf{DTMC})
  - Each process has a local boolean variable \( x_i \)
  - Token in place \( i \) if \( x_i = x_{i+1} \)
  - Basic step of process \( i \):
    - if \( x_i = x_{i+1} \) make a uniform random choice as to the next value of \( x_i \)
    - otherwise set \( x_i \) to the current value of \( x_{i+1} \)
  - Allow to start in any state (\textbf{MDP})

- In the PRISM language:

```
module process1
    x1 : bool;
    [step]  x1=x2    -> 0.5 : x1'=0 + 0.5 : x1'=1;
    [step]  !(x1=x2) -> x1'=x2;
endmodule

module process2 = process1 [x1=x2, x2=x3] endmodule
    ...
module processN = process1 [x1=xN, x2=x1] endmodule
```
Results: Herman’s protocol

- \( P_{\leq 1}(\text{stable}) \): min probability of reaching a stable state is 1
- \( E_{\leq 2}(\text{stable}) \): max expected time (number of steps) to reach a stable state, assuming initially \( K \) tokens and \( N \) processes:
Israeli-Jalfon’s self-stabilising protocol

- Asynchronous ring of $N$ processes (MDP)
- Each process has a local boolean variable $q_i$
  - token in place $i$ if $q_i$=true
  - process is active if and only if has a token
  - basic step of (active) process: uniform random choice as to whether to move the token to the left or right

- In the PRISM language:

```prism
module process1
  s1 : bool; // dummy variable
  [] (q1=1) -> 0.5 : (q1'=0) & (qN'=1) + 0.5 : (q1'=0) & (q2'=1);
endmodule

module process2 = process1 [s1=s2, q1=q2, q2=q3, qN=q1] endmodule

module processN = process1 [s1=sN, q1=qN, q2=q1, qN=qN-1] endmodule
```
Results: Israeli-Jalfon’s protocol

- $P_{\geq 1}(\text{stable})$: min probability of reaching a stable state is 1
- $E_{\geq 2}(\text{stable})$: max expected time (number of steps) to reach a stable state, assuming initially $K$ tokens and $N$ processes:
Beauquier, Gradinariu and Johnen’s self-stabilising protocol

- Asynchronous ring of $N$ ($N$ odd) processes (MDP)
  - Each process has two boolean variables: $d_i$ and $p_i$ where:
    - if $d_i = d_{i-1}$ process $i$ is said to have a deterministic token
    - if $p_i = p_{i-1}$ process $i$ is said to have a probabilistic token
    - stable states are those where there is only one probabilistic token
    - process is active if and only if has a deterministic token
  - Basic step of (active) process $i$:
    - negate $d_i$ and if $p_i = p_{i-1}$, then set $p_i$ uniformly at random
  - In the PRISM language:

```plaintext
module process1
    d1 : bool; p1 : bool;
    [] d1=d3 & p1=p3 -> 0.5 : (d1'=d1) & (p1'=p1) + 0.5 : (d1'=d1) & (p1'=p1);
    [] d1=d3 & !p1=p3 -> (d1'=d1);
endmodule

module process2 = process1 [d1=d2, d2=d3, p1=p2, p2=p3] endmodule

; ;

module processN = process1 [d1=dN, d2=d1, p1=pN, p2=p1] endmodule
```
Results: Beauquier, Gradinariu and Johnen’s protocol

- $P_{\text{stable}}$: min probability of reaching a stable state is 1
- $E_{\text{stable}}$: max expected time (number of steps) to reach a stable state, assuming initially $K$ tokens and $N$ processes:
Case study: IPv4 Zeroconf protocol

- IPv4 ZeroConf protocol [Cheshire, Adoba, Guttman'02]
  - New IETF standard for dynamic network self-configuration
  - Link-local (no routers within the interface)
  - No need for an active DHCP server
  - Aimed at home networks, wireless ad-hoc networks, hand-held devices
  - “Plug and play”

- Self-configuration
  - Performs assignment of IP addresses
  - Symmetric, distributed protocol
  - Uses random choice and timing delays
IPv4 Zeroconf Standard

- Select an IP address out of 65024 at random
- Send a probe querying if address in use, and listen for 2 seconds
  - If positive reply received, restart
  - Otherwise, continue sending probes and listening (2 seconds)
- If K probes sent with no reply, start using the IP number
  - Send 2 packets, at 2 second intervals, asserting IP address is being used
  - If a conflicting assertion received, either:
    - defend (send another asserting packet)
    - defer (stop using the IP address and restart)
Will it work?

- Possible problem...
  - IP number chosen may be already in use, but:
    - Probes or replies may get lost or delayed (host too busy)

- Issues:
  - Self-configuration delays may become unacceptable
    - Would you wait 8 seconds to self-configure your PDA?
  - No justification for parameters
    - for example K=4 in the standard

- Case studies:
  - DTMC and Markov reward models, analytical [BvdSHV03,AK03]
  - TA model using UPPAAL [ZV02]
  - PTA model with digital clocks using PRISM [KNS03]
The IPv4 Zeroconf protocol model

• Modelled using Probabilistic Timed Automata (with digital clocks)

• Parallel composition of two PTAs:
  - one (joining) host, modelled in detail
  - environment (communication medium + other hosts)

• Variables:
  - \( K \) (number of probes sent before the IP address is used)
  - the probability of message loss
  - the number of other hosts already in the network
Modelling the host
Modelling the environment
Expected costs

- Compute minimum/maximum expected cost accumulated before obtaining a valid IP address?

- Costs:
  - \textbf{Time should be costly}: the host should obtain a valid IP address as soon as possible
  - \textbf{Using an IP address that is already in use should be very costly}: minimise probability of error

- Cost pair: \((r,e)\)
  - \(r=1\) (\(t\) time units elapsing corresponds to a cost of \(t\))
  - \(e=10^{12}\) for the event corresponding to using an address which is already in use
  - \(e=0\) for all other events
Results for IPv4 Zeroconf

- Sending a high number of probes increases the cost
  - increases delay before a fresh IP address can be used
- Sending a low number of probes increases the cost
  - increases probability of using an IP address already in use
- Similar results to the simpler model of [BvdSHV03]
Case Study: Bluetooth protocol

- **Short-range low-power wireless protocol**
  - Personal Area Networks (PANs)
  - Open standard, versions 1.1 and 1.2
  - Widely available in phones, PDAs, laptops, ...

- **Uses frequency hopping scheme**
  - To avoid interference (uses unregulated 2.4GHz band)
  - Pseudo-random frequency selection over 32 of 79 frequencies
  - Inquirer hops faster
  - Must synchronise hopping frequencies

- **Network formation**
  - Piconets (1 master, up to 7 slaves)
  - Self-configuring: devices *discover* themselves
  - Master-slave roles
States of a Bluetooth device

- Master looks for device, slave listens for master
- **Standby**: default operational state
- **Inquiry**: device discovery
- **Page**: establishes connection
- **Connected**: device ready to communicate in a piconet
Why focus on device discovery?

• **Performance of device discovery crucial**
  - No communication before initialisation
  - First mandatory step: *device discovery*

• **Device discovery**
  - Exchanges information about slave clock times, which can be used in later stages
  - Has considerably higher power consumption
  - Determines the speed of piconet formation
Frequency hopping

- **Clock** CLK, 28 bit free-running, ticks every 312.5\(\mu\)s
- **Inquiring device** (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):

\[
\text{freq} = \left[ \text{CLK}_{16-12} + k + (\text{CLK}_{4-2,0} - \text{CLK}_{16-12}) \mod 16 \right] \mod 32
\]
**Frequency hopping sequence**

\[ \text{freq} = ((\text{CLK}_{16-12} + k + (\text{CLK}_{4-2,0} - \text{CLK}_{16-12})) \mod 16) \mod 32 \]

- Two trains (=lines)
- \( k \) is offset that determines which train
- Swaps between trains every 2.56 sec
- Each line repeated 128 times
Sending and receiving in Bluetooth

- **Sender**: broadcasts inquiry packets, sending according to the frequency hopping sequence, then listens, and repeats

- **Receiver**: follows the frequency hopping sequence, own clock
  - Listens continuously on one frequency
  - If hears message sent by the sender, then replies on the same frequency
  - Random wait to avoid collision if two receivers hear on same frequency
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 scale constants (approximate results)
  - Cannot omit subactivities, otherwise oversimplification

- **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
  - 17,179,869,184 possible initial states

- **But is a realistic future ubiquitous computing scenario!**
What about other approaches?

• Indeed, others have tried...
  - network simulation tools (BlueHoc)
  - analytical approaches

• But
  - simulations obtain averaged results, in contrast to best/worst case analysis performed here
  - analytical approaches require simplifications to the model
  - it is easy to make incorrect probabilistic assumptions, as we can demonstrate

• There is a case for all types of analyses, or their combinations...
Lessons learnt...

- **Must optimise/reduce model**
  - Assume negligible clock drift, obtain a DTMC
  - Manual abstractions, combine transitions, etc
  - Divide into 32 separate cases
  - Success (*exhaustive analysis*) with one/two replies

- **Observations**
  - Work with *realistic constants*, as in the standard
  - Analyse v1.2 and 1.1, confirm 1.1 slower
  - Show best/worst case values, can pinpoint scenarios which give rise to them
  - Also obtain *power consumption* analysis

- **Performance of device discovery crucial**
  - No communication before initialisation
  - First mandatory step: *device discovery*
Max time to hear is 2.5716 sec, in 921,600 possible initial states, (Min 635 µs)

Cumulative: assume uniform distribution on states when receiver first starts to listen
Time to hear 2 replies

- Max time to hear is 5.177 sec (16,565 slots), in 444 possible initial states
- Cumulative (derived): assumes time to reply to 2\textsuperscript{nd} message is independent of time to reply to 1\textsuperscript{st} (incorrect, compare with exact curve obtained from model checking)
Case Study: FireWire Protocol

- **FireWire (IEEE 1394)**
  - one of fastest standards, high data rate
  - multimedia data
  - originally by Apple, mid-90s
  - winner of 2001 PrimeTime Emmy Engineering Award
  - no requirement for a single PC (acyclic topology, not tree)
  - “plug and play”

- **Initial configuration**
  - involves leader election
  - symmetric, distributed protocol
  - uses electronic coin tossing and timing delays: PTA model
Typical FireWire Configuration
FireWire Initial Configuration
FireWire Root Contention
FireWire Root Contention
FireWire Analysis

- **Real-time properties**
  - analysed by Vandraager and Stoelinga
  - used the UPPAAL model checker
  - shown correct wires longer than standard

- **Probabilistic analysis**
  - used UPPAAL & PRISM model checkers [KNS03, DNK02]
  - timing delays taken from standard
  - established that root contention resolved with probability 1
  - also considered expected time to root contention
  - a peculiarity found... (conjectured by Stoelinga)

- **Further analyses** at various levels of abstraction, see special issue on FireWire

FireWire: Analysis Results
Unfair coin gives advantage!

Expected time

Probability of flipping "tails"
Related projects

- **FORWARD (this case study, see ISOLA’04)**
  - Performance modelling of MAC layer of Bluetooth
  - Security analysis of Bluetooth

- **Modelling and verification of mobile ad hoc network protocols**
  - Modelling language with mobility and randomisation
  - Model checking algorithms & techniques
  - Tool development & implementation
  - Modelling timing properties of AODV [FMOODS’05]

- **Modelling biological processes**
  - Biochemical reactions: eukaryotic cell cycle, ERK pathway, FGF pathway
  - Integrative Biology: modelling cancer, crypt development in colorectal cancer
Extending PRISM with mobility

- Models in PRISM
  - are described in reactive modules
    - extend with mobility, dynamic topology
    - extend with geographical positioning
    - extend with context-awareness
  - are finite-state, static and often huge
    - verification support for compositionality, abstraction
    - techniques for infinite state systems
    - combine with simulation-based methods

- Specifications
  - are temporal logic based:
    - add location-awareness
    - more expressive logics?
Challenges for future

• Exploiting structure
  - Abstraction, data/equivalence quotient, (de)compositionality...
  - Parametric probabilistic verification?
• Proof assistant for probabilistic verification?
• Approximation methods?
• Efficient methods for continuous models
  - Continuous PTAs? Continuous time MDPs? LMPs?
• More expressive specifications
  - Probabilistic LTL/PCTL*/mu-calculus?
• Real software, not models!

• More applications
  - Quantum cryptographic protocols
  - Mobile ad hoc network protocols
For more information...

J. Rutten, M. Kwiatkowska, G. Norman and D. Parker

**Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems**

P. Panangaden and F. van Breugel (editors), CRM Monograph Series, vol. 23, AMS
March 2004

www.cs.bham.ac.uk/~dxp/prism/

- Case studies, statistics, group publications
- Download, version 2.1 (2000 downloads)
- Unix/Linux, Windows, Apple platforms
- Publications by others and courses that feature PRISM...
Collaborators, contributors – thanks!