Computational Stochastic Modelling for Large-scale Systems: Methods and Applications

Rashid Mehmood
School of Engineering
Swansea University
R.Mehmood@swansea.ac.uk

17 June 2010
Modelling and Analysis of Distributed Systems, SICSA
University of Stirling

The Drivers

- Pervasive/Ubiquitous Environments
- Ambient Intelligence
- Autonomic/Self-configurable Systems
- Infrastructure/Computing Clouds
- Smaller and faster Computational devices and capabilities
- Virtualisation
- Human Computer Interaction
- Semantic Web, Knowledge representation and inference
- Social networks

The System

- Inputs: Designer, Operator, User and Machine Interfaces
  - Human-like languages with formal reasoning
  - Data from sensors and databases
  - How reliable is the system (failure chance)
  - My privacy policies/preferences are...
- Mathematical and Computational System Models
  - Intelligence and/or Analysis, e.g. Data fusion
- Actuators/Control Units, Visualisation etc

Computational Stochastic Modelling

- Computational Stochastic Modelling (CSM)
  - Why? A Design, Analysis and Management tool?
  - What is it? What it involves? What methods are available?
  - Where do Probability, Markov Chains, and Simulations fit in?
  - Why do we need Parallel and Grid Computing?
  - Do we need numerical methods and matrix computations?
  - Do we need compact data structures?
  - Do we need efficient algorithms?
  - Do we need computational strategies? time-space trade-off etc.

- Applications in computing and communication systems

Outline

- Computational Stochastic Modelling
- Solution Methods and Results
- Applications
- Overview of Research and Activities

Stochasticity, Modelling and Simulation

- Why consider Stochasticity
  - most real-life systems are inherently stochastic
    - Economy, telecommunication and road networks, demographic data
    - The living organisms and their interaction with the environment
- Modelling and Simulation
  - Important design, analysis and management tools
  - Are modelling and simulation different? Hybrid?
  - Stochastic processes and simulations
  - Closed form solutions and analytic tractability
  - Mean value analysis and computation of probability distributions
- Computational Engineering is the common element
  - Data structures, algorithms and computational strategies

Discrete State Approach

- Design, Analysis, and Management tools
  - Physical Experiments and Empirical Studies
  - Stochastic Processes
  - Stochastic Simulations, Discrete Event
  - Discrete State Models
  - Markov Chains, CTMCs (continuous time)
  - Markov Decision Processes

- Focus will be Discrete State Modelling
  - Behaviour of physical systems
    - a set of states
    - state to state transitions

- Markov Chains: here, the main stochastic model

Computational Stochastic Modelling

- Model Checking
  - Specify all the states a system could enter
  - Explore binary answers
  - Software verification community

- Stochastic (Probabilistic) Model Checking [Kwiatkowska]
  - Specify states, and transition probabilities
  - Explore state probabilities

- Computational Stochastic Modelling
  - Augment the above with computations and simulations, wherever possible
  - Trade with computations, wherever possible

CSM: The Three Phases

- Higher Level Specification
  - Why do we need it?
  - Allows clarity, abstraction, convenience, robustness
  - GUIs could be an example
  - SysML in Systems Engineering
  - UML, Queuing Networks, Stochastic Petri Nets, …
  - Formal Methods: Process Algebras and Logics
    - System and Property Specification (CSL, PCTL, …)
  - Allows questions such as “would M4 be congested today”

- The intermediate level
  - Generation of state space and transition rates
  - matrix generation from the formalism

CSM: The Three Steps

- Final Phase: Transient and Steady State Solution
- – Differential equation for transient
  - $\partial x(t) / \partial t = x(t)A$
- – Steady-state solution
  - $Ax = 0, \sum x_i = 1, x = \lim_{t \to \infty} x(t)$
- – Optimisation Problems (MDPs)
  - $Ax \geq 0$ or $Ax \leq 0$

- Problem: State-space explosion
- – numerical solution of $Ax = 0$
- – Matrix computations involving large matrices and vectors
- – The matrix A is [sparse]

State-Space Explosion

- solution of $Ax = 0$
  - requires storage of the matrix A and the vector(s) x

- Numerical methods: need fast but low in storage
  - Direct and iterative methods
    - Gaussian elimination (fill-in)
    - Jacobi, Power, Gauss-Seidel (slow, low in storage)
    - Krylov subspace methods (fast, high in storage)

- Storage: need compact storage but fast access
  - Implicit methods: BDD-based storage
  - explicit methods: methods from linear algebra community
  - Parallel and out-of-core techniques
    - seek storage and CPU alternatives

Markov Process

- A Markov Process is
  - a Stochastic Process: $\{x(t) \mid t \in T\}$
  - for any $t_0 < t_1 < t_2 < \ldots < t_k < t$
  - the conditional distribution of $X(t)$ depends only on $x(t_k)$
  - (the values taken by $x(t)$ are process states)

- Intuitively…

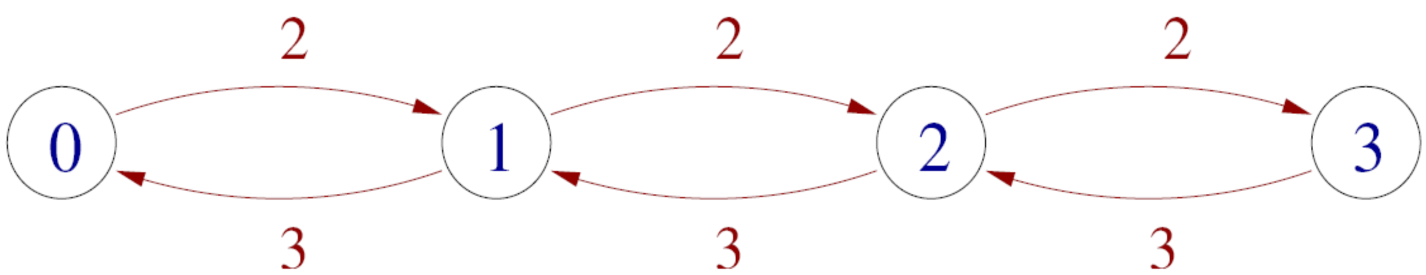
- Mathematically:
  - $P[x(t) \leq x \mid x(t_k) = x_k, x(t_{k-1}) = x_{k-1}, \ldots, x(t_0) = x_0]$
  - equals $P[x(t) \leq x \mid x(t_k) = x_k]$

Markov Chain

- MP with discrete states -> Markov chain
- Continuous-time and discrete-time
- A CTMC:
  - a set of states, S
  - a transition rate matrix, $\mathrm{R}: S \times S \longrightarrow \mathbb{R}$
  - state-to-state transition: if $r_{i,j} > 0$
  - the mean sojourn time for state i is $1/E(i)$
  - $E(i) = \sum_{j \in S,\, j \neq i} r_{i,j}$
  - Transition probability: state i to j within t time units: $1 - e^{-E(i)t}$
  - $q_{i,i} = -E(i)$
  - CTMC generator sparse matrix Q

Transition Diagram and Matrix

- A Simple Example
  - Markov Chain: Web Server or Road Network or …



$$\begin{pmatrix} -2 & 2 & 0 & 0 \\ 3 & -5 & 2 & 0 \\ 0 & 3 & -5 & 2 \\ 0 & 0 & 3 & -3 \end{pmatrix}$$

Steady State Solution

- Numerical Steady State Solution
- Probabilities of system to be in a particular state
  - in the long run

$$-2\pi_0 + 3\pi_1 = 0,$$

$$2\pi_0 - 5\pi_1 + 3\pi_2 = 0,$$

$$2\pi_1 - 5\pi_2 + 3\pi_3 = 0,$$

$$2\pi_2 - 3\pi_3 = 0,$$

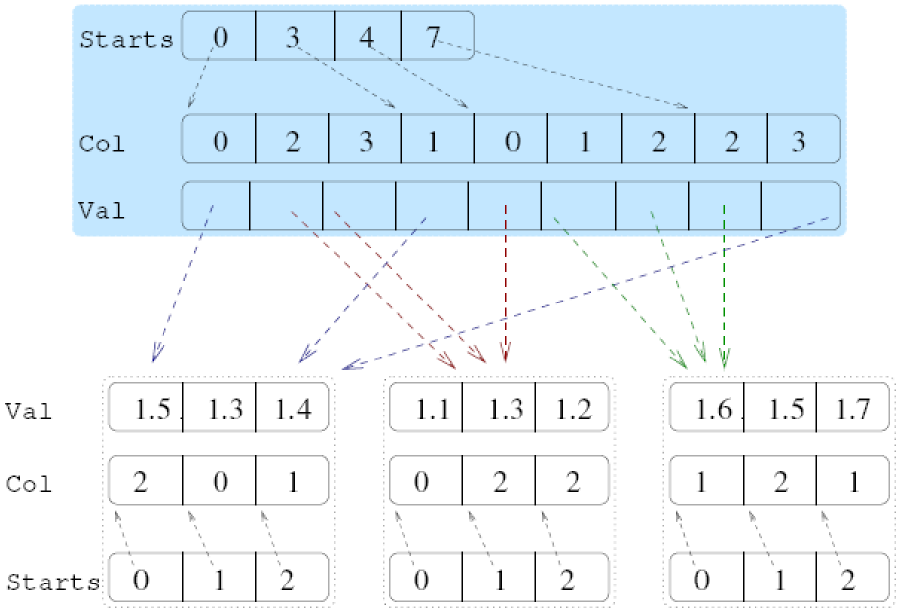
$$\pi_0 + \pi_1 + \pi_2 + \pi_3 = 1,$$

$$\pi = [\frac{27}{65}, \frac{18}{65}, \frac{12}{65}, \frac{8}{65}]$$

The Symbolic CTMC Representation

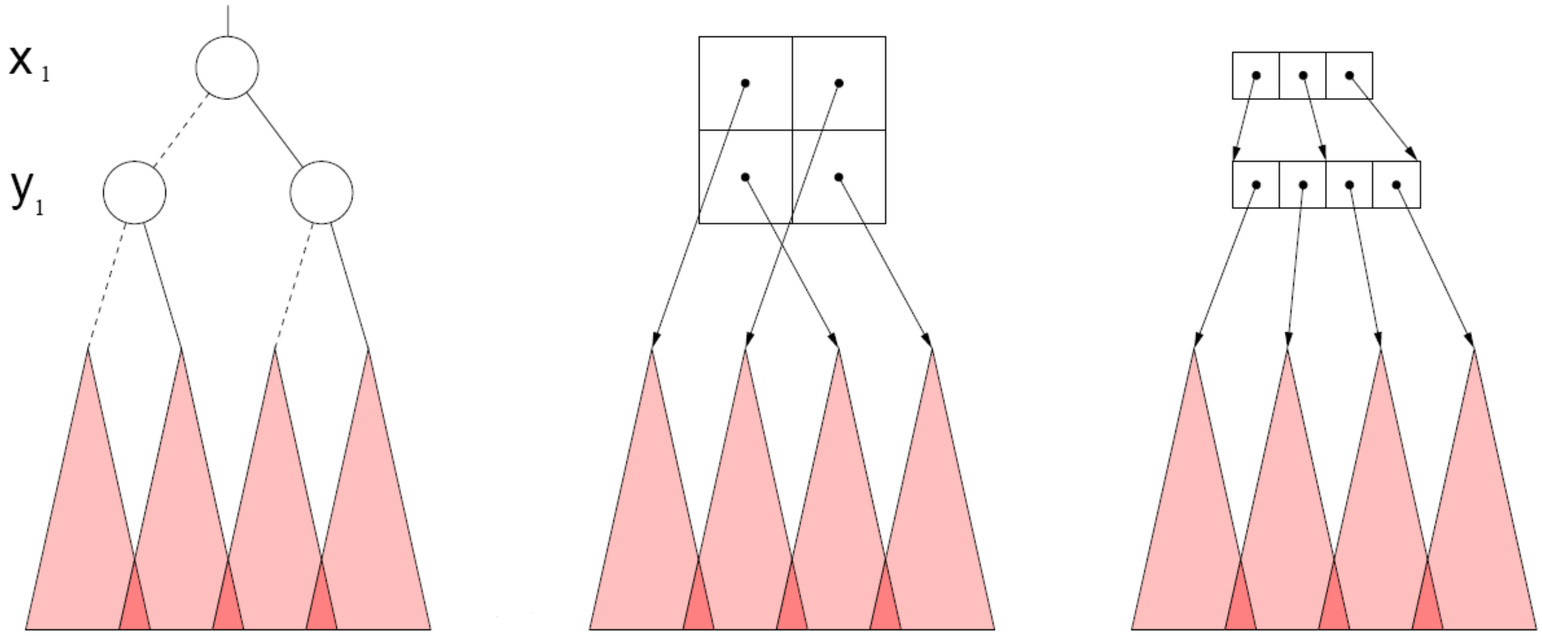
- Modified Multi-Terminal Binary Decision Diagrams (MTBDDs) [Meh04]
- Decompose matrix into blocks
- Store blocks individually
- Store the high-level information about each block

0	0	1.5				1.1	0	0	1.1	0	0
1.3	0	0				0	0	1.3	0	0	1.3
0	1.4	0				0	0	1.2	0	0	1.2
			0	0	1.5						
			1.3	0	0						
			0	1.4	0						
1.1	0	0	0	1.6	0	0	1.6	0			
0	0	1.3	0	0	1.5	0	0	1.5			
0	0	1.2	0	1.7	0	0	1.7	0			
						0	1.6	0	0	0	1.5
						0	0	1.5	1.3	0	0
						0	1.7	0	0	1.4	0



Multi Terminal BDDs [MPK03, Par03]

- Decision Diagrams -> Matrix -> Sparse Matrix



Case Studies

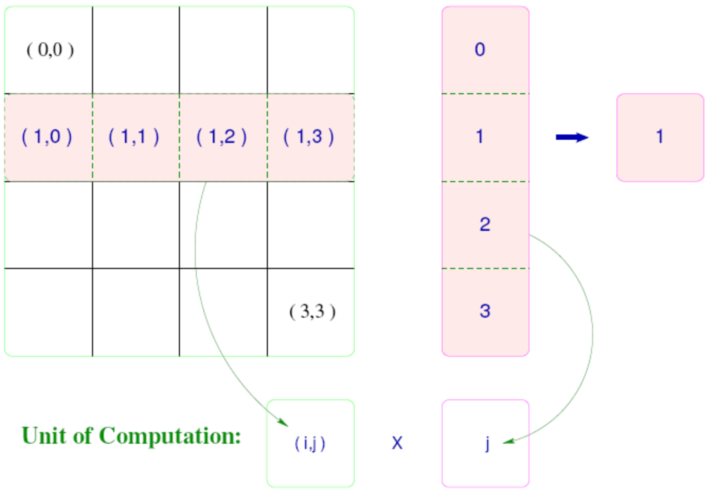
- Generated using PRISM Model Checker
- Polling (Cyclic Server Polling System) [IT90]
  - k stations or queues
  - and a server, which polls the queues in a cycle looking for jobs
- FMS (Flexible Manufacturing System) [CT93]
  - three machines process different types of parts
  - k: maximum number of parts each machine can handle
- Kanban manufacturing System [CT96]
  - a total of four machines
  - k: max. number of jobs in a machine at one time

Comparison of Storage Schemes

k	States (n)	Off-diagonal nonzeros (a)	a/n	Memory for Matrix (MB)				Vector (MB)
				MSR	*Ind. MSR*	*Comp. MSR*	*MTBDDs*	
				FMS models				
6	537,768	4,205,670	7.82	50	24	17	4	4
10	25,397,658	234,523,289	9.23	2,780	1,366	918	137	194
13	216,427,680	2,136,215,172	9.87	25,272	12,429	8,354	921	1,651
14	403,259,040	4,980,958,020	12.35	58,540	28,882	19,382	1,579	3,077
15	724,284,864	9,134,355,680	12.61	107,297	52,952	35,531	2,676	5,526
				Kanban models				
4	454,475	3,979,850	8.76	47	23	16	1	3.5
6	11,261,376	115,708,992	10.27	1,367	674	452	6	86
9	384,392,800	4,474,555,800	11.64	52,673	25,881	17,435	99	2,933
10	1,005,927,208	12,032,229,352	11.96	141,535	69,858	46,854	199	7,675
				Polling System				
15	737,280	6,144,000	8.3	73	35	24	1	6
21	66,060,288	748,683,264	11.3	8,820	4,334	2,910	66	504
24	603,979,776	7,751,073,792	12.8	91,008	44,813	30,067	144	1,136
25	1,258,291,200	16,777,216,000	13.3	196,800	96,960	65,040	317	1,190

An MVP-based Computation

- Partitioning
  - A: $B \times B$ square blocks (not necessary)
  - x: B blocks with n/B entries each
- A unit of Computation: A sub-MVP
  - matrix block $\times$ vector block $(A_{ij} \times X_j)$



Serial Block Jacobi Algorithm

ser_block_Jac($\breve{A}$, d, b, x, P, $n[\,]$, ε) {

1. $\underline{\texttt{var}}\ \tilde{x},\ Y,\ k \leftarrow 0,\ \texttt{error} \leftarrow 1.0,\ i,\ j,\ p$
2. $\underline{\texttt{while}}(\ \texttt{error} > \varepsilon)$
3. $\quad k \leftarrow k + 1$
4. $\quad \underline{\texttt{for}}(\ 0 \leq i < P)$
5. $\quad\quad Y \leftarrow B_i$
6. $\quad\quad \underline{\texttt{for}}(\ 0 \leq j < P)$
7. $\quad\quad\quad Y \leftarrow Y - \breve{A}_{ij} X_j^{(k-1)}$
8. $\quad\quad \underline{\texttt{for}}(\ 0 \leq p < n[i])$
9. $\quad\quad\quad X_i^{(k)}[p] \leftarrow D_i[p]^{-1} Y[p]$
10. $\quad$ compute $\texttt{error}$
11. $\quad X_i^{(k-1)} \leftarrow X_i^{(k)}$

}

Out-of-Core Algorithm

Integer constant: B (*number of blocks*)
Semaphores: S_1, S_2: occupied
Shared variables: R_0, R_1 (*To read matrix A blocks into RAM*)

Disk-IO process

Local variables: i, j, $t = 0$
`while` not converged
    `for` $i \leftarrow 0$ to $B - 1$
        `for` $j \leftarrow 0$ to $B - 1$
            `if` not an *empty* block
                disk_read (A_{ij}, R_t)
            Signal(S_1)
            Wait(S_2)
            $t = (t + 1) \ mod \ 2$

Compute process

Local variables: i, j, $t = 0$
`while` not converged
    `for` $i \leftarrow 0$ to $B - 1$
        `for` $j \leftarrow 0$ to $B - 1$
            Wait(S_1)
            Signal(S_2)
            `if` $j \neq B - 1$
                `if` not an *empty* block
                    sub-MVP($A_{ij}X_j$, R_t)
            `else`
                Update X_i
                check for convergence
            $t = (t + 1) \ mod \ 2$

Out-of-Core on Single Machine

Model	k	States (n)	a/n	Times		Iterations
				Per iteration (seconds)	Total (hr:min:sec)	
FMS	11	54,682,992	9.5	51.6	23:16:39	1624
	12	111,414,940	9.7	170	84:54:20	1798
	13	216 427 680	9.9	327	179:34:39	1977
	14	403,259,040	10.03	1984	–	> 50
	15	724,284,864	10.18	6312	–	> 50
Kanban System	7	41,644,800	10.8	18.9	4:12:38	802
	8	133,865,325	11.3	139	38:34:21	999
	9	384,392,800	11.6	407	136:54:37	1211
	10	1,005,927,208	11.97	1424	566:49:52	1433
Polling System	22	138,412,032	11.8	143	1:28:11	37
	23	289,406,976	12.3	264	2:47:12	38
	24	603,979,776	12.8	460	4:51:20	38
	25	1,258,291,200	13.3	1226	13.16:54	39

A Parallel Jacobi Algorithm for p

```
par_block_Jac( $\breve{A}_p$, $D_p$, $B_p$, $X_p$, $T, N_p$, $\varepsilon$ ) {

var $\tilde{X}_p$, $Z$, $k \leftarrow 0$, error $\leftarrow 1.0$, $q$, $h$, $i$
while( error $> \varepsilon$)
    $k \leftarrow k + 1$;  $h \leftarrow 0$
    for( $0 \leq q < T; q \neq p$)
        if( $\breve{A}_{pq} \neq \mathbf{0}$)
            send( request$_{X_q}$, $q$);  $h \leftarrow h + 1$
    $Z \leftarrow B_p - \breve{A}_{pp} X_p^{(k-1)}$
    while( $h > 0$)
        if( probe( message) )
            if( message = request$_{X_p}$)
                send( $X_p, q$)
            else
                receive( $X_q, q$);  $h \leftarrow h - 1$
                $Z \leftarrow Z - \breve{A}_{pq} X_q^{(k-1)}$
    serve( $X_p$, request$_{X_p}$)
    for( $0 \leq i < N_p$)
        $X_p^{(k)}[i] \leftarrow D_p[i]^{-1} Z[i]$
    compute error collectively                    }
```

Parallel Execution on 24 Nodes

k	States (n)	MB/Node	Time: Iteration (seconds)	Time: Total (hr:min:sec)	Total Iterations
			FMS Model		
12	111,414,940	170	6.07	3:40:57	2184
13	216 427 680	306	13.50	8:55:17	2379
14	403,259,040	538	25.20	18:02:45	2578
15	724,284,864	1137	48.47	37:26:35	2781
			Kanban System		
7	41,644,800	53	1.73	33:07	1148
8	133,865,325	266	5.27	2:02:06	1430
9	384,392,800	564	14.67	7:03:29	1732
10	1,005,927,208	1067	37.00	21:04:10	2050
			Polling System		
22	138,412,032	328	2.60	44:31	1027
23	289,406,976	667	5.33	1:36:02	1081
24	603,979,776	811	11.60	3:39:38	1136
25	1,258,291,200	1196	23.97	7:54:25	1190

Applications

Computing and Communication Systems

Ongoing Work

- Location based security, services, visualisation
- Multimedia Ad hoc Networks (Analysis and Design)
- Middleware for Healthcare, e-learning [AM07]
- Risk Management for ITS and Healthcare sectors [AMW10]
- Traffic Virtual Reality simulator [Ayres08, AM09]
- Integration of communication and traffic simulations/models
- LocPriS: A Security and Privacy Preserving Location Based Services Development Framework
- Vehicular Networks - Modelling, Simulations, Feature Interaction
- Managing Pervasive Environments: Mobile e-learning and Intra-Vehicular [AGM10, ANGM10]

Pervasive System Management

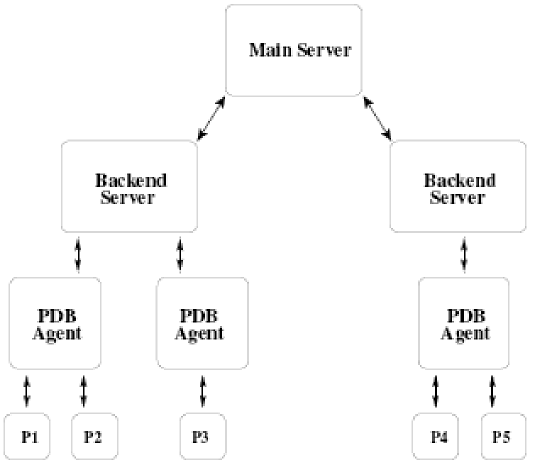
- Developing and managing distributed systems is hard
  - autonomy implies nondeterminism
  - synchronisation problems and race conditions
  - deadlocks
- Developing debugging, and management tools are even harder
  - networks are unpredictable
  - system behaviour may not be reproducible
  - race conditions are possible even in network absence
  - the probe effect
  - global state may not be visible
- System Virtualisation: Xen (Cambridge)
  - Pervasive Debugger including Middleware

Pervasive Debugging [HHHM04, MCHH05]

- Extension to large-scale distributed systems
  - a hierarchical and scalable architecture
    - decompose and distribute the problem
    - e.g. a distributed assertion
    - $G = f_0 \wedge f_1 \cdots \wedge f_{99}$ replaced by
    - $G = G_0 \wedge G_1 \cdots \wedge G_9$ and $G_i = f_0 \wedge f_1 \cdots \wedge f_9$
    - localise tasks and traffic
  - use cluster/grid of machines running Xen
  - rollback machines to realise globally consistent states

PDB Architecture

- PDB Agent
    - sits within a virtual machine
    - instruments, controls processes
- Backend Server
    - manages multiple PDB agents
    - responsible to Main Server
- Intermediate Server
    - generalised Backend Server
    - manages multiple Backends
- Main Server
    - manages Intermediate or Backend Servers
- Further details in [MCHS05]



PDB Functionality

- A standard debugger interface
  - stop/inspect/go
  - `breakpoint` and `watchpoint` events
  - can also generate `blocking` and `unblocking` events
  - other events, e.g. `send`, `recv`, `timer`, are planned
  - `timer` can be used to detect e.g. component failures

- Verifying program behaviour
  - compose events to specify distributed assertions
  - e.g. "*no more than one node believes itself to be the leader*"
  - can define/create consumers
  - can subscribe to interesting events
  - invoke arbitrary consumers if program behaviour is violated

Computation Is Alive

Assertion: *computation is alive*

1. forever
2. *sleep*()
3. set `waiting`
4. *pick_left_chopstick*()
5. *pick_right_chopstick*()
6. reset `waiting`
7. *eat*()
8. *put_left_chopstick*()
9. *put_right_chopstick*()

```
def phil_is_blocked(phil):
    return phil.blocked()
    and phil.waiting == 1

def deadlocked():
    return phil_is_blocked(phil1)
        and phil_is_blocked(phil2)
        and phil_is_blocked(phil3)

def philosophers_are_alive():
    return ¬ deadlocked()

def ring_email_cons():
    ring the bell
    email Alice

Alive = Assertion(philosophers_are_alive)
MyConsumer = Consumer(ring_email_cons)
MyConsumer.subscribe(Alive)
```

Computations Are Useful

see [MC05]

Assertion: *computations are useful*

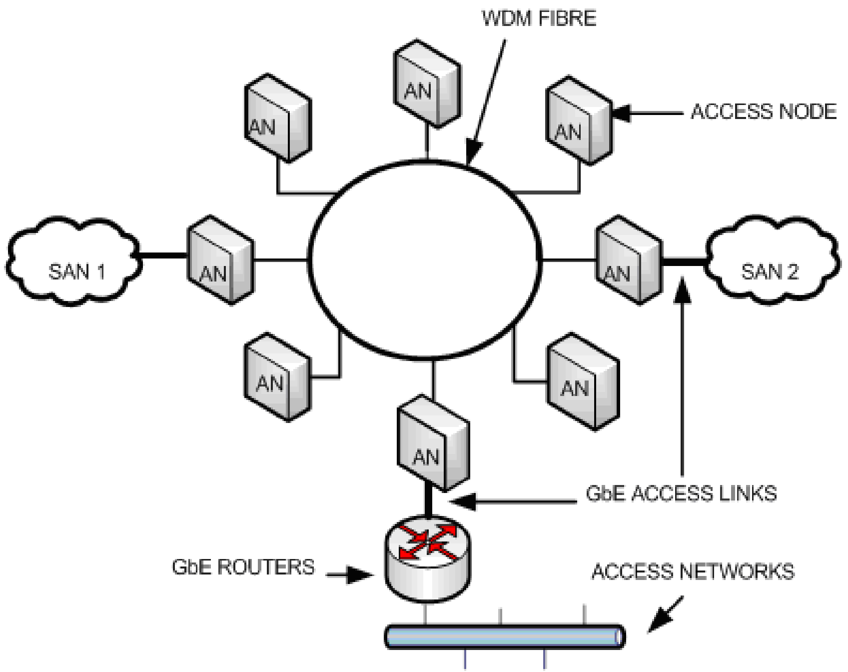
```
while error > ε
    for j = 1 to p
        if A_ij is not a zero block
            send request for x_j
    accumulate sub-MVP A_ii x_i
    while true
        wait for incoming message
        if message
            if a request for x_i from proc j
                send x_i to proc j
            else
                receive x_j from proc j
                acc. sub-MVP A_ij x_j
        if my comps. finished, break
    serve any remaining requests
    update x_i
    collectively calculate error
```

```
global.error_t = 10
global.count = 0
def count_set():
    if p.error ≥ global.error_t:
        global.count += 1
        global.error_t = p.error
    else:
        global.count = 0
    if global.count == 100:
        Email Alice

WP = WatchPoint(p.error)
MyCons = Consumer(count_set)
MyCons.subscribe(WP)
```

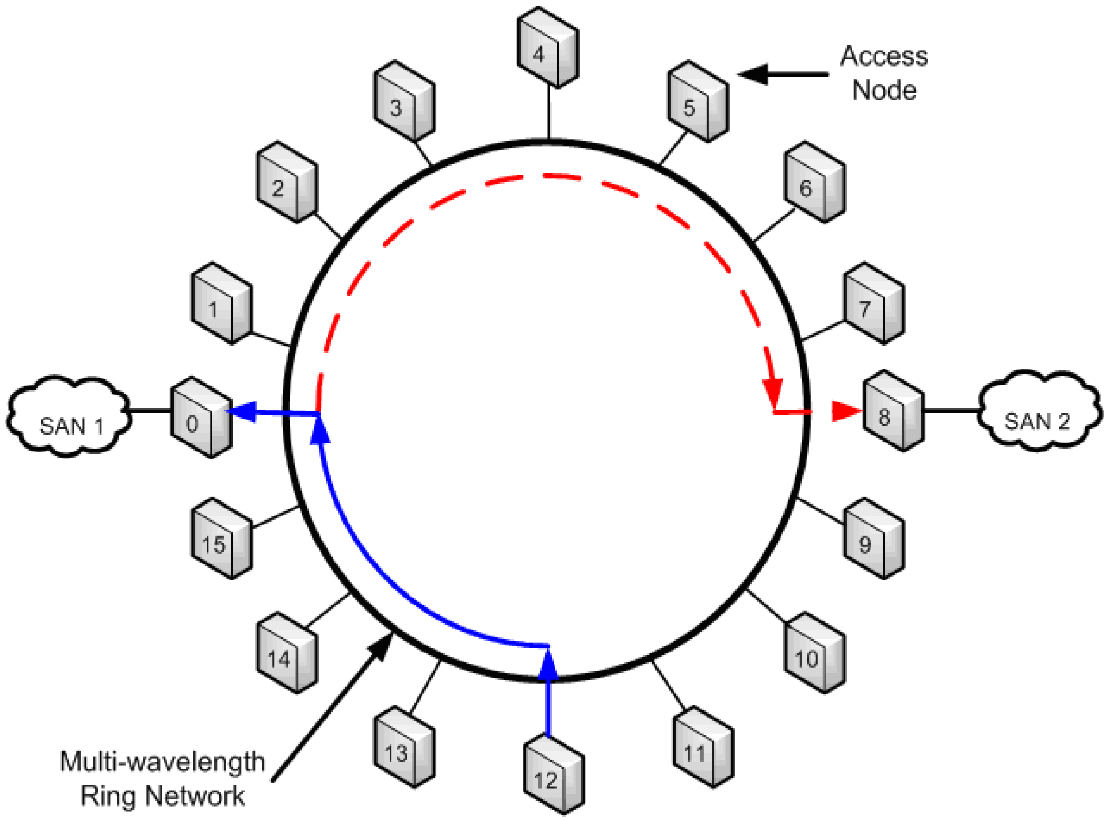
Optical Networks [PME07]

- Metropolitan Area Networks (MAN)
- WDM Ring
- Symmetric Traffic
- Asymmetric
- Poisson process
- Self-Similar
- Multimedia
- SAN extensions



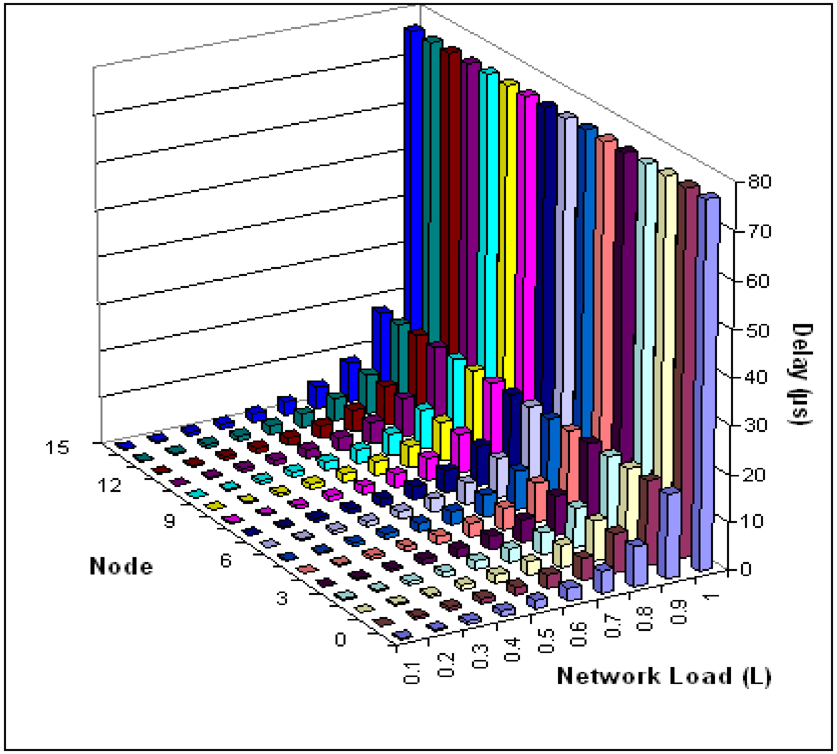
SAN Mirroring

- for backup over large distances [PME07, EPME07]



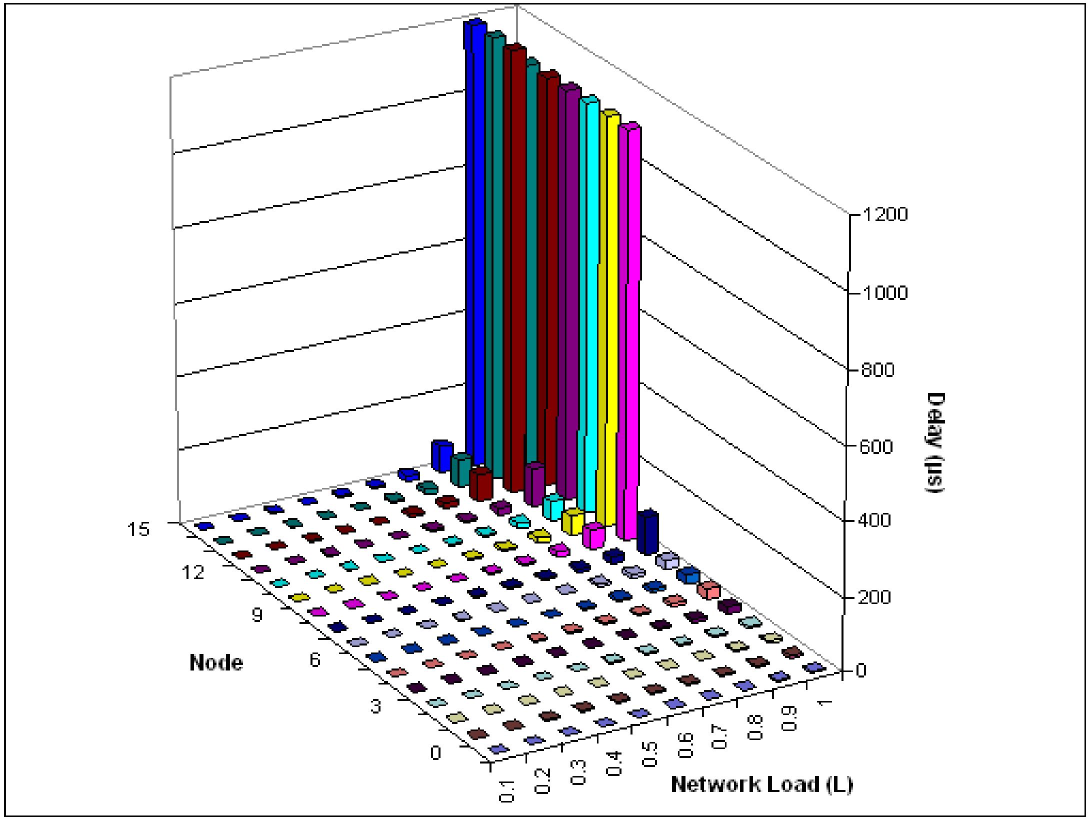
Applications Considered (MAN)

- Symmetric Traffic



Applications Considered (MAN)

- Asymmetric Traffic



Wireless Systems[AM07,AM08]

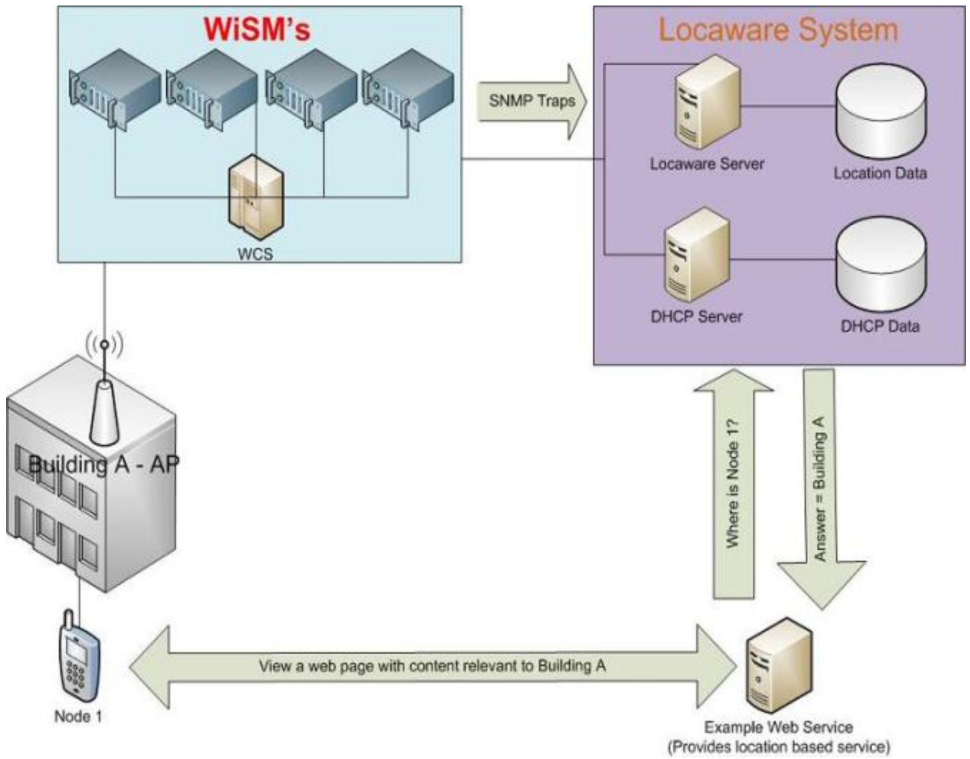
- Wireless spectrum is limited
  - demand for bandwidth and latency seems unlimited
  - Traffic is asymmetric and unpredictable
- We need optimum control of network
  - Best use of network resources including bandwidth
  - Maximum subscribers per service per unit area
  - May be try minimum acceptable QoS
- Focus: a resource allocation scheme
  - multimedia traffic in 3G environment
  - improve QoS
  - reduce blocking and dropping probabilities
  - reduced packet losses and queuing time

Infrastructure and Ad hoc Networks

- Multimedia performance over MAN, LAN, University Campus
- Ad hoc Networks [MAF09]
  - Performance Analysis
  - Routing protocols
  - Cross-layer optimisation
  - OPNET Software
- Vehicular Ad hoc Networks (VANET)
- VGNets (scientific applications) [MN07]

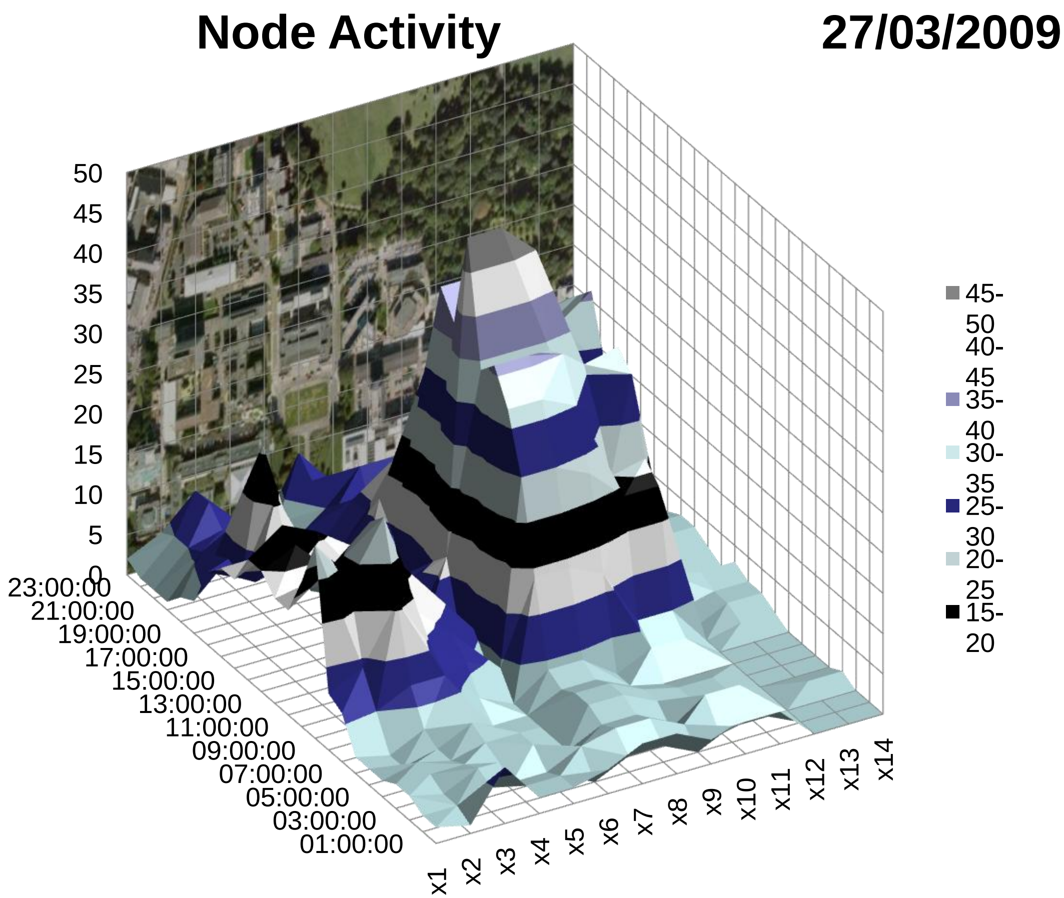
Location Aware (JANET) [AMMR09]

- Location Based Security and Services

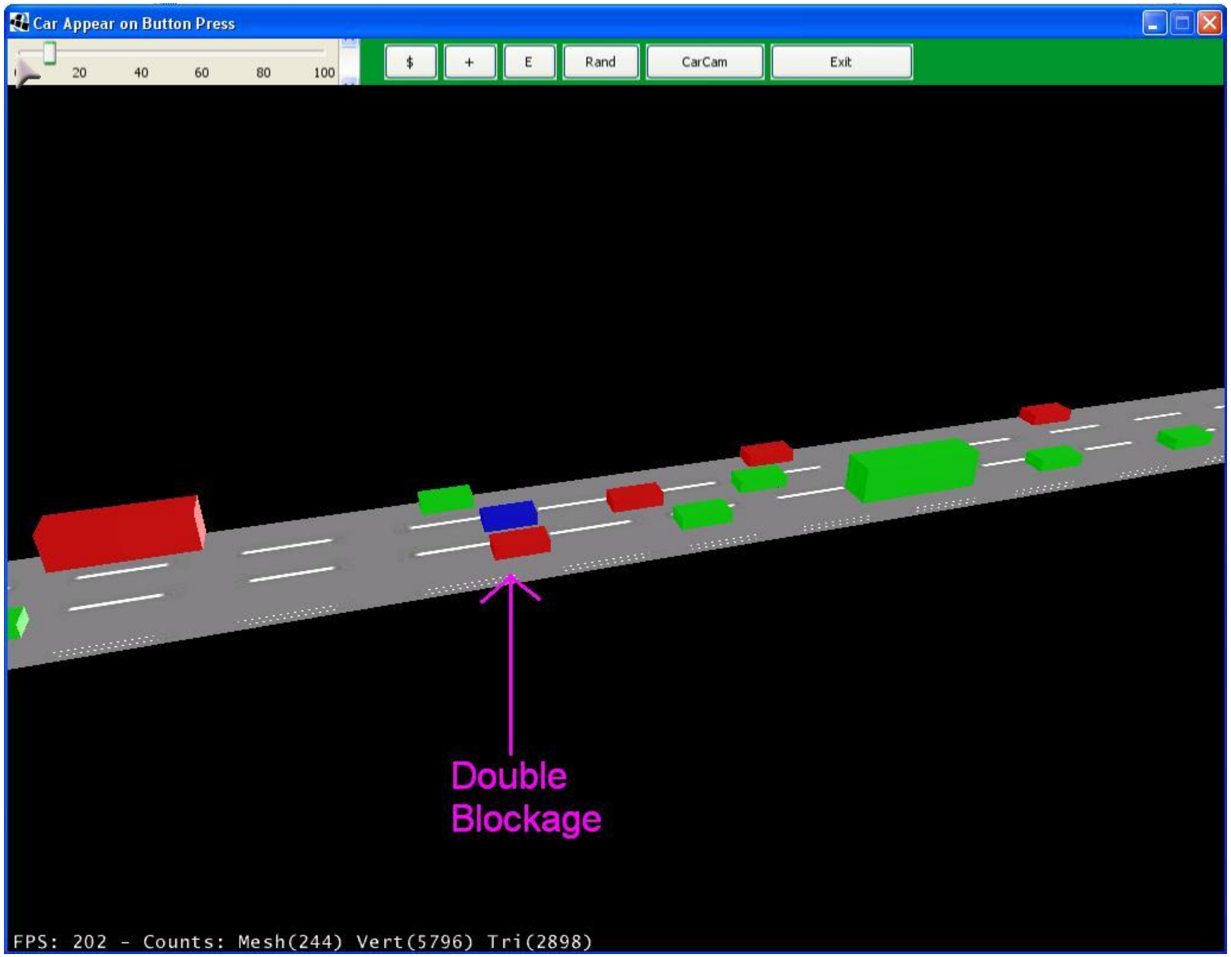


Location Aware (JANET)

Location Aware (JANET)



Road Traffic VR Simulator



Digital Economy

- Research Councils Programme
  - Initially funding to form research Clusters
  - Feasibility Studies in ICT developments
  - allow early user adoption
  - Three core area
  - Healthcare, Transport and Creative Industries
- MRSN
  - Many-core and Reconfigurable Supercomputing Network

Digital Economy

- Opportunities and Challenges in the Digital Economy
  - an Agenda for the Next Generation Internet
  - Infrastructure and Services

Digital Economy

- SIMM
  - Services for Intelligent Mobility Management in the Digital Economy
- Inclusive DE
  - An Inclusive Digital Economy supporting Older and Disabled People and other Digitally Disenfranchised groups

Digital Economy

- IMDE
  - Innovative Media for a Digital Economy
- e-Health+
  - Citizen-driven Information for Healthcare and Wellbeing

Conferences/Meetings

- EuropeComm 2009, London
  - Comm, ITS, IHS, and Business Models
- EuropeComm 2011, Budapest
- Nets4Cars 2010, Newcastle
- Nest4Cars 2011, Germany
- Digital Intelligence – 14th KES 2010, Cardiff

Conclusion

- Formal methods theory has developed significantly over the past two decades
- State space explosion is a problem but we can solve larger systems
- Automatic extraction of models from real world systems is required
  - Real time monitoring and management is possible
- We need more expressive, powerful formal methods
- Techniques to solve larger and larger systems in shortest possible time
  - Powerful hardware platforms are emerging: Many-cores, GPUs

That is All…

- Thank You.