

Experiences with Open Overlays: A Middleware Approach to Network Heterogeneity

*Paul Grace, Danny Hughes, Barry Porter,
Gordon Blair, Geoff Coulson, Francois Taiani*



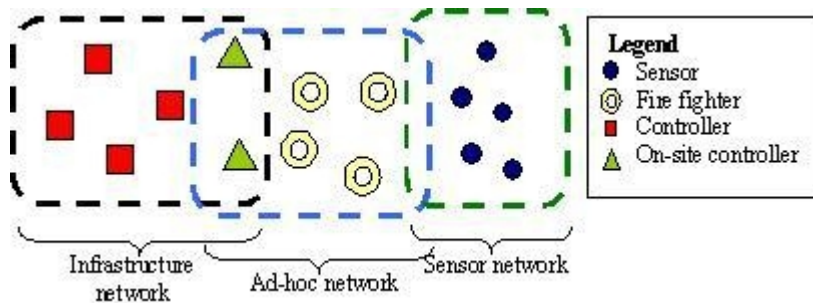
Introduction

- Increasing range of interworking network technologies
 - Applications face extreme heterogeneity
- Contribution: Embrace network overlays as the central enabling technology of middleware
 - Virtualise the network resource
 - Co-existence of physical and virtual overlays
 - layering (re-use) of overlays
- Detailed evaluation of the open overlays software framework
 - Adaptation case-study
 - Development and deployment experiences

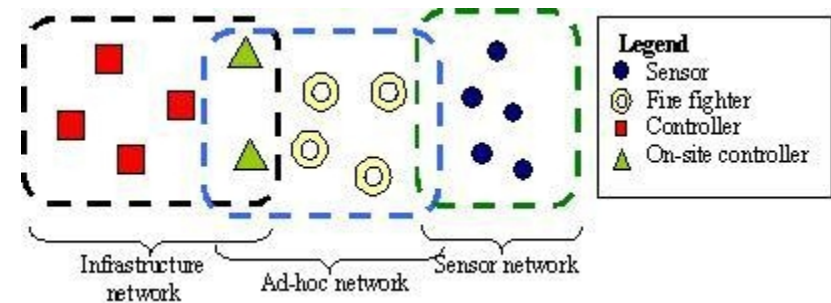


Motivation: Extreme Heterogeneity

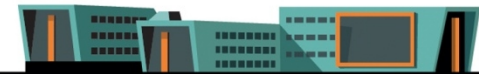
Emergency Response



Sensor Grids (Flood Prediction)



- Diversity in terms of end-systems and networked infra-structures:
 - high-speed LANs, lower-speed WANs, infrastructure-based wireless networks, ad-hoc wireless networks
 - cluster systems through to miniature sensor devices



Responses to Heterogeneity

- Add more and more features to existing middleware
 - E.g. Mobile extensions
 - Leads to bloated middleware
- Specialised middleware per application domain
 - E.g. Middleware for sensor networks, mobile middleware, ...
 - Successful, but limits future interoperability
- Configurable frameworks
 - Tailored to the needs of the application and the operational domain(s)

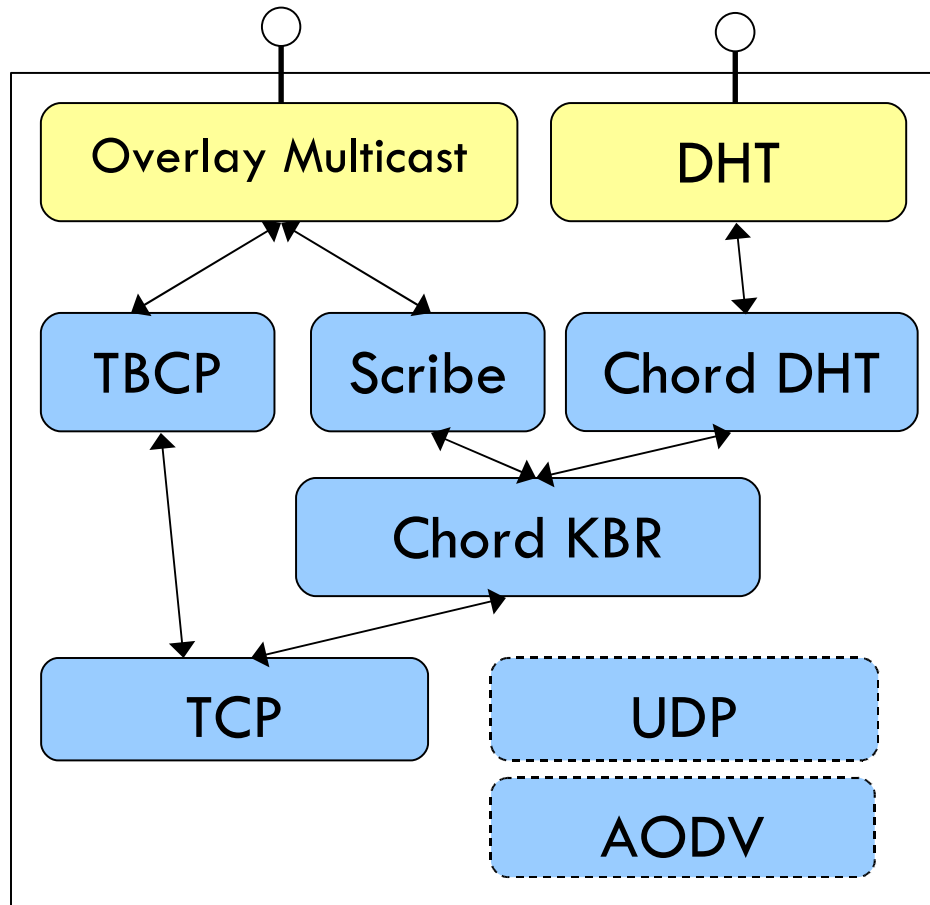


Open Overlays

- ❑ Central element of contemporary middleware
- ❑ Configurable and reconfigurable software framework
 - i. Flexible *Virtualisation* of the network resource
 - ii. *Co-existence* of multiple physical & virtual network abstractions
 - iii. *Layering* of virtual network abstractions



The Overlays Software Framework



Interface plug-ins
(virtualisation)

Overlay plug-ins

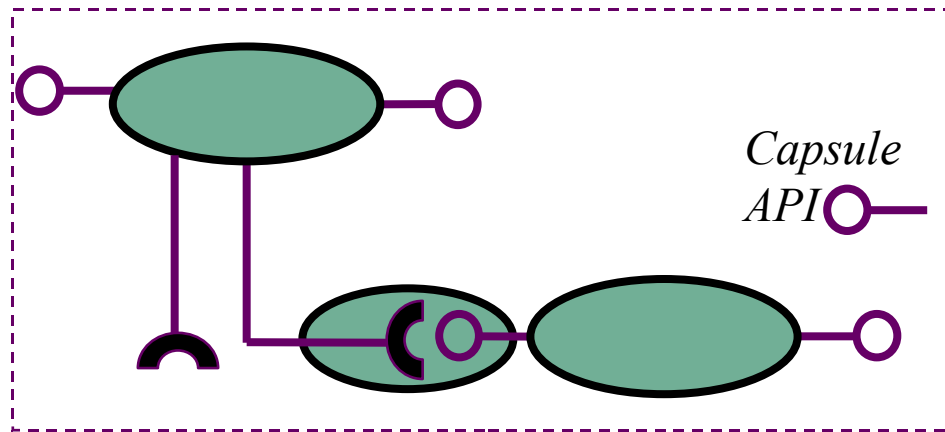
b) Layering for richer services

c) Co-existence



The OpenCOM component model

- central concepts:
component | *capsule* | *interface* | *receptacle* | *binding*
- Benefits of component-engineering at the systems software level
 - High-level abstraction for development and management
 - Third-party re-use
 - Flexible configuration and dynamic reconfiguration



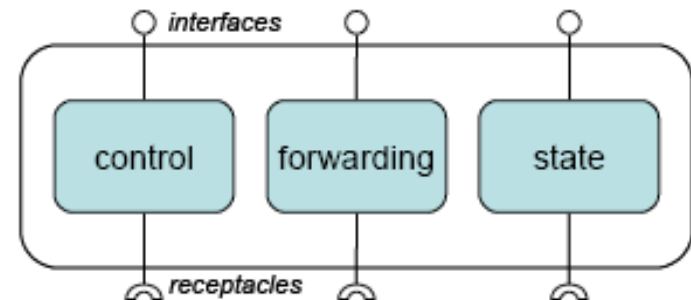
Developing Individual Overlay Plug-ins

Software Architecture

- Reusability
- Software Patterns
- Multi-party development and integration
- Coordinated adaptation
- Verification

Component Frameworks

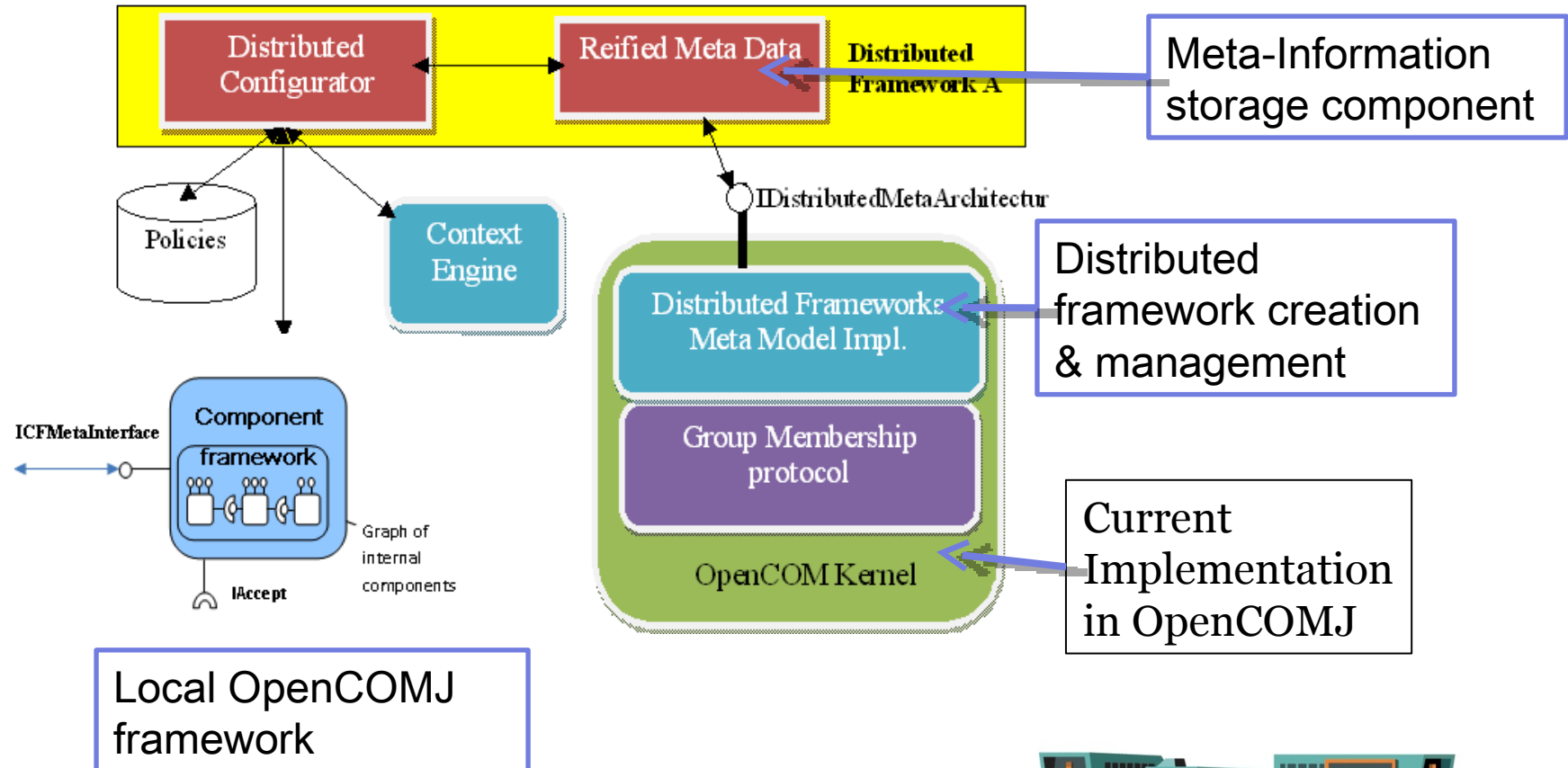
Overlay Pattern



- Topology
- Behaviour
 - Create, Join, Leave, Send, Receive, Deliver



Reconfiguring a distributed software framework



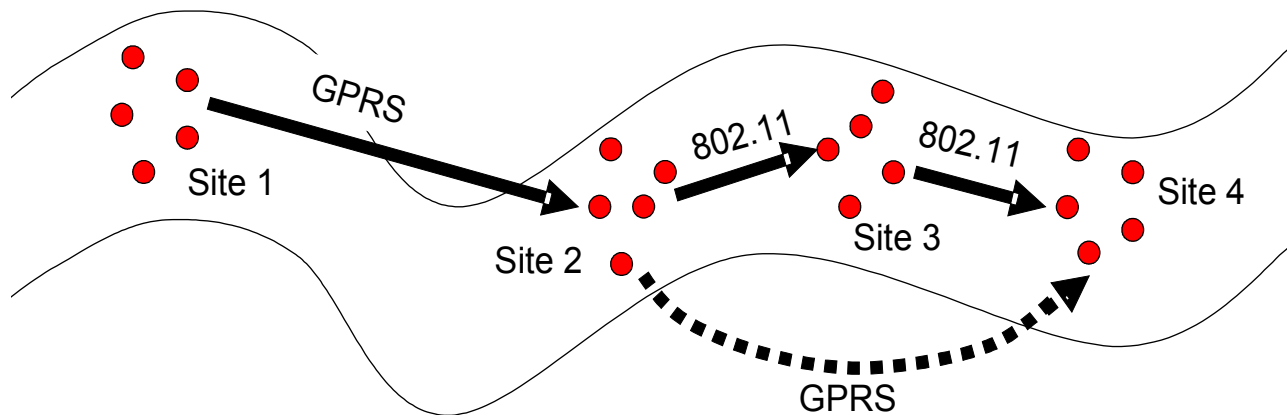
Evaluating the Open Overlays Concept

1. Reconfigurability
2. Configurability
3. Generality
4. Ease of use
5. Resource Overhead

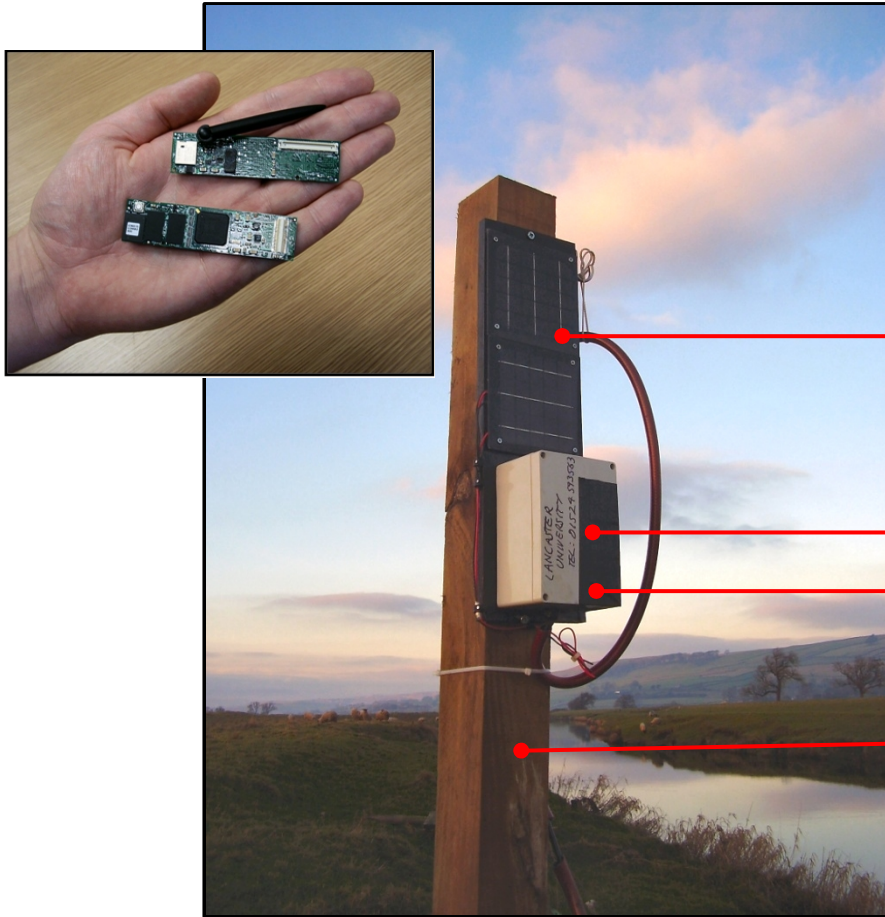


Adaptation Case-study

- ▶ Collaboration with hydrologists from Lancaster's Environmental Science department
 - ▶ Predict flooding in a river valley in NW England



Deployment



Power:

- 4W Solar Array (15CM * 30CM)
- 84WH Gel Battery.

Processors:

- 400MHz XScale, 64MB RAM, 16MB Flash ~ 2W.
- 16MHz AVR, 4KB RAM, 128KB Flash ~ 0.1W.
- 8MHz PIC, < 1KB RAM, 1KB Flash ~ 0.01W.

Networking:

- Bluetooth & 802.11b.
- A subset have GPRS and 433MHz Radios

Sensors:

- Analogue depth, conductivity and flow sensors.

Flood-WSN Profile

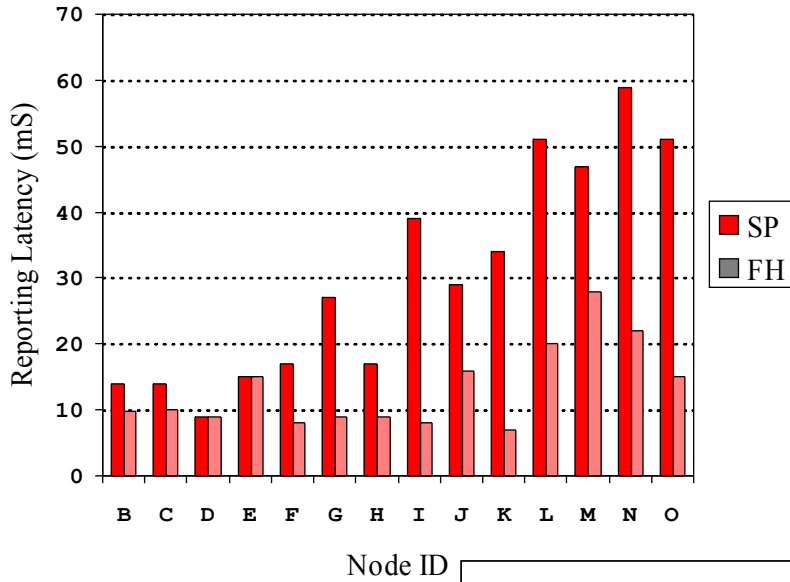
- ❑ At the physical network level (Bluetooth or WiFi)
 - ❑ Bluetooth used in quiescent conditions, WiFi in flooding

	Throughput	Power Draw	Range
Bluetooth	786Kbps	0.4W typical	up to 200M
WiFi	11Mbps	2.9W typical	up to 1.2KM

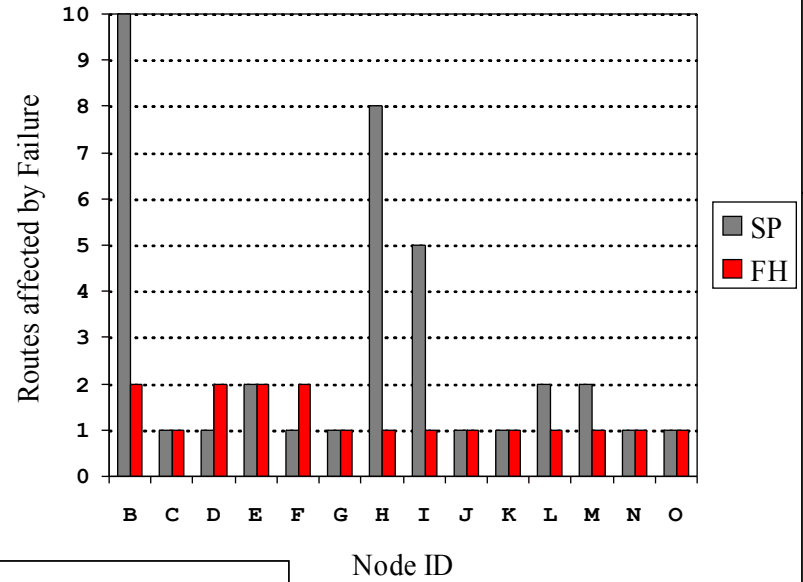
- ❑ At the data routing level
 - ❑ Shortest-path spanning tree (SP)
 - ❑ Less power
 - ❑ Fewest-hop spanning tree (FH)
 - ❑ Better performance, Better resilience



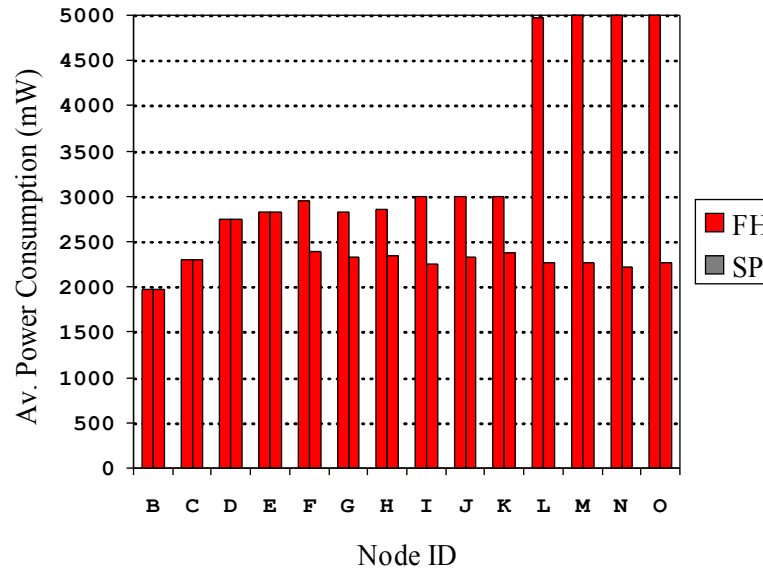
Spanning Tree Latency



Spanning Tree Resilience



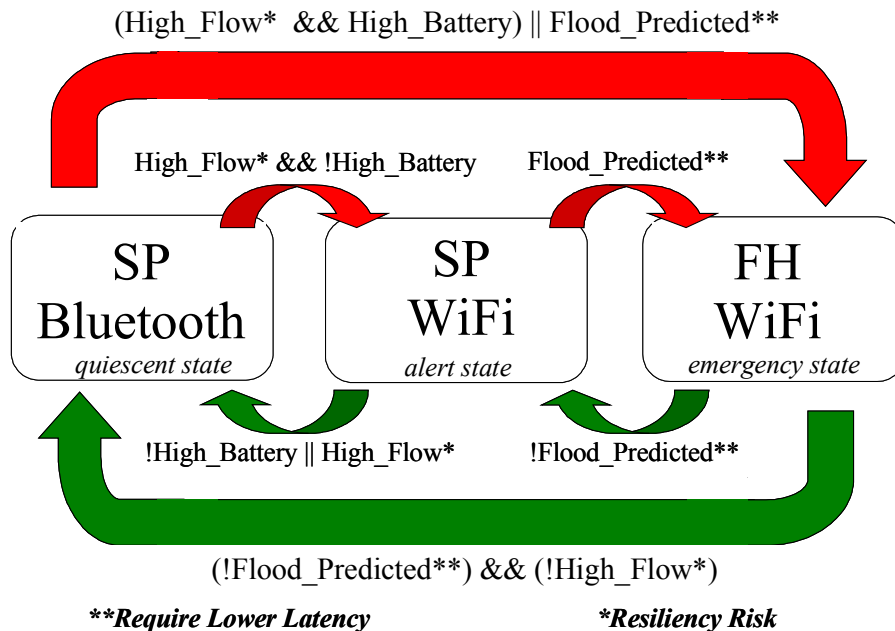
Spanning Tree Power Consumption



Experiments to inform adaptation policies



Reconfiguration Policies

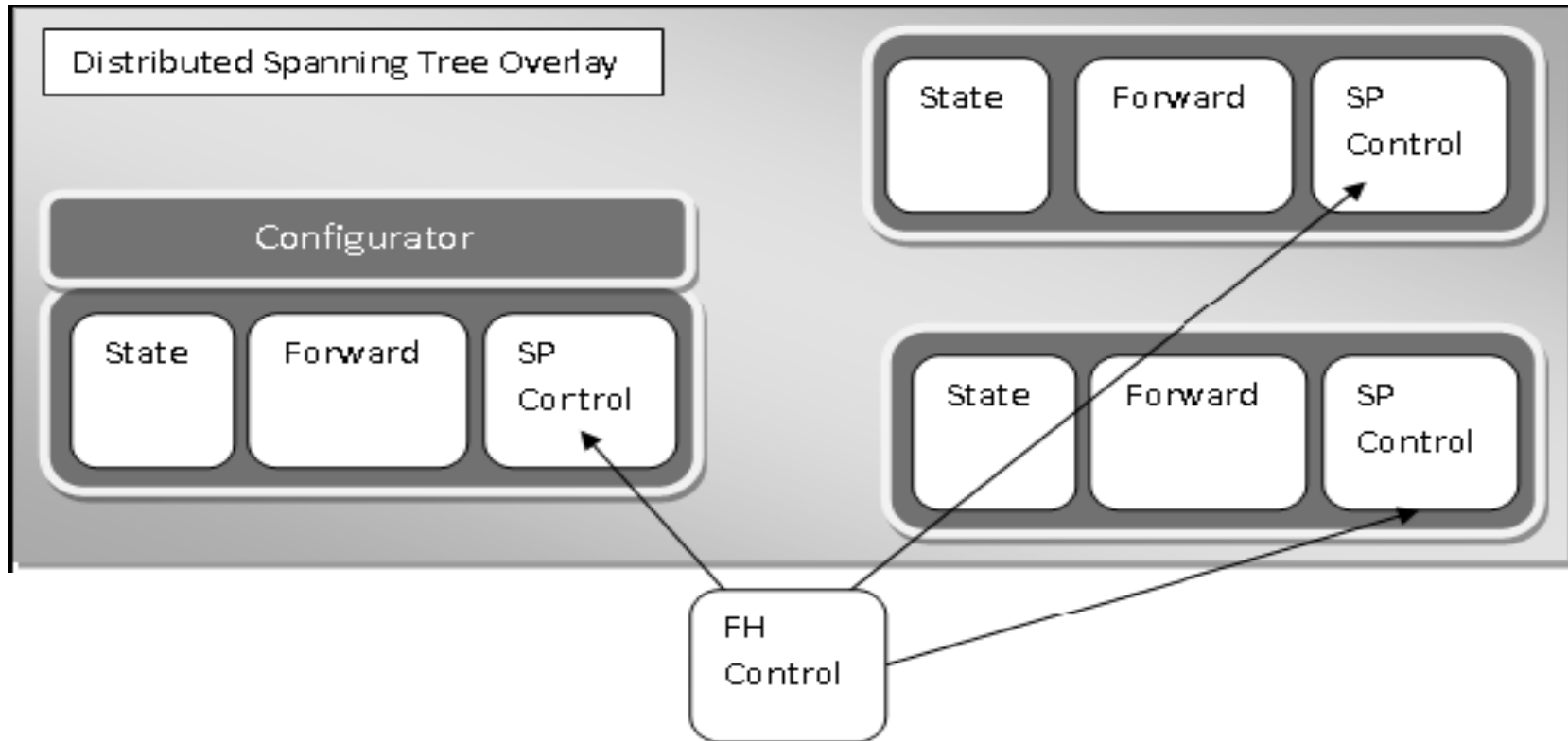


```

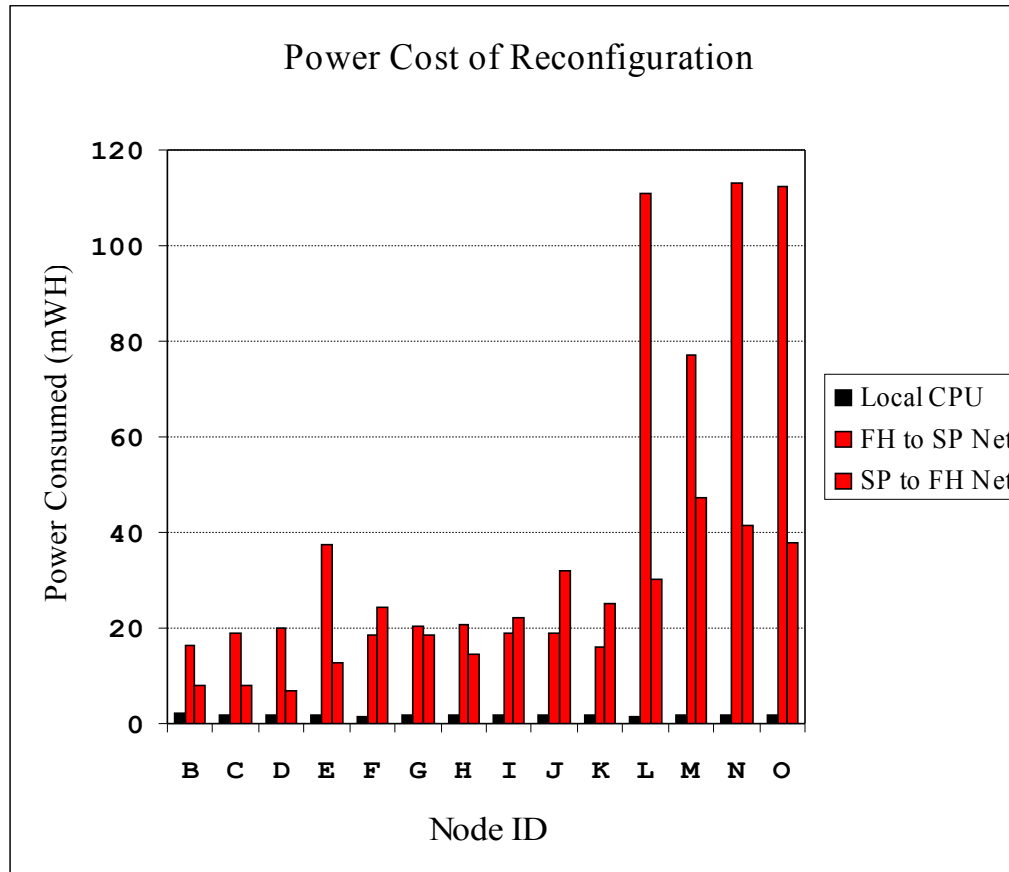
if (High_Flow && High_Battery
    || Flood_Predicated)
  replace Bluetooth with WiFi
  replace ST.sp with ST.fh
  
```



Example Co-ordinated reconfiguration (SP to FH)



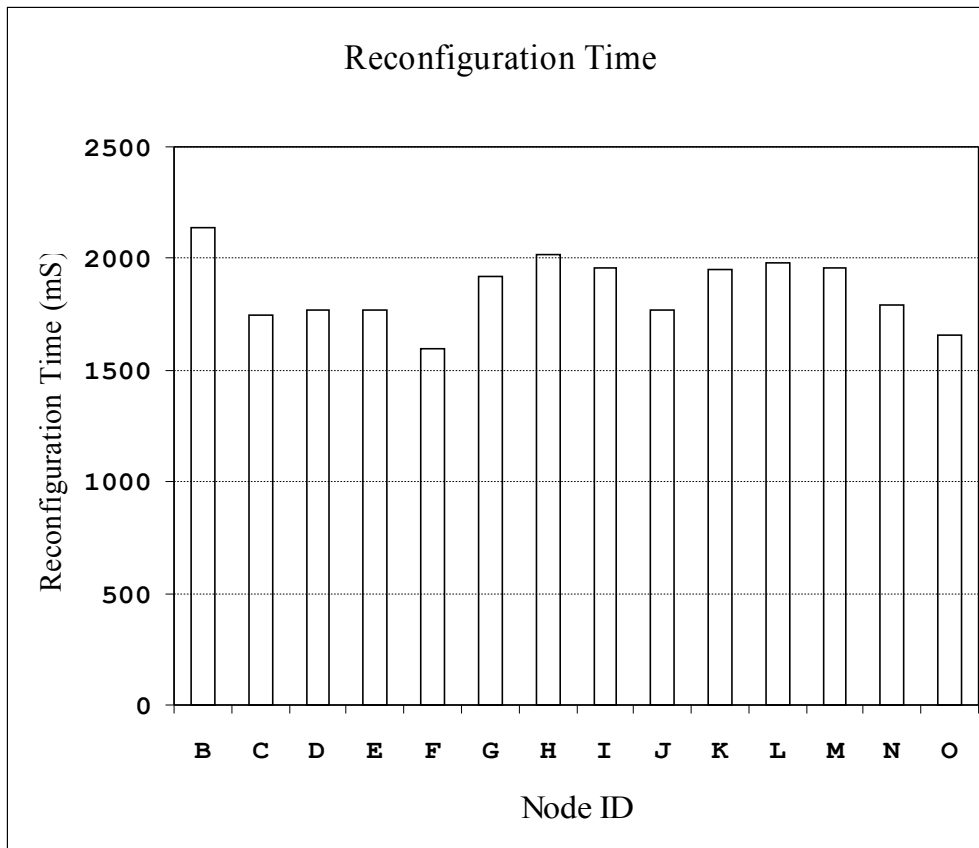
Reconfiguration Costs (Power)



0.05% of the
battery
capacity of a
GridStix



Reconfiguration Costs (time)



- 1873 ms av. Compared to 53 ms on PCs
- Within reporting time range



Spanning Tree Overhead (per node)

	Shortest Path (SP)		Fewest Hops (FH)	
	<i>Size</i>	<i>LoC</i>	<i>Size</i>	<i>LoC</i>
Control	7.3KB	124	<i>re-used</i>	<i>re-used</i>
State	2.5KB	24	<i>re-used</i>	<i>re-used</i>
Forward	6.3KB	346	6.5KB	352



Generality

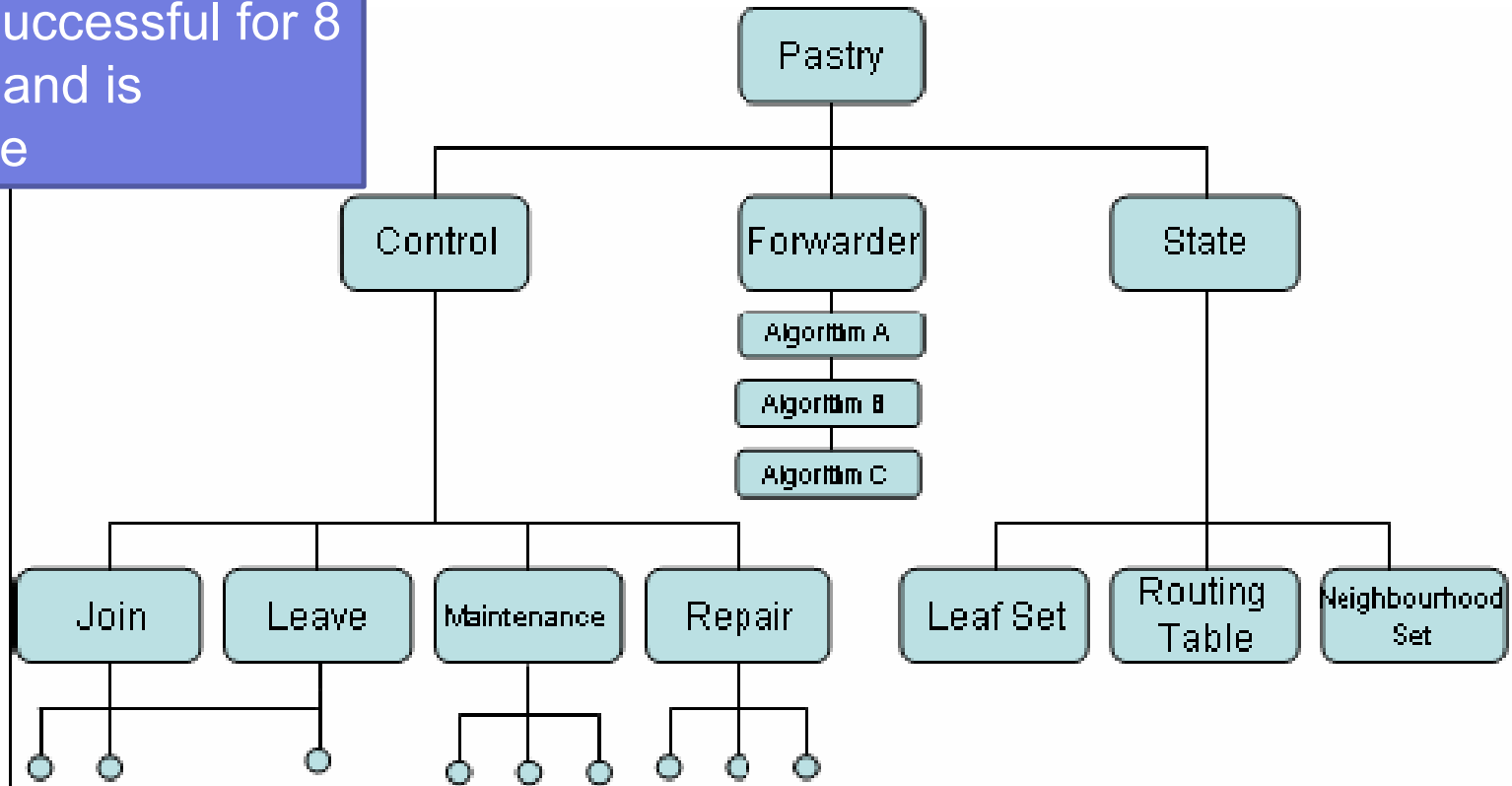
Overlay Name	Description and configurability options
Chord KBR	A KBR overlay based on Chord
DHT	Data storage overlay
Pastry KBR	A KBR overlay based on Pastry
Failure Monitor	Monitoring overlay; detects and disseminates node failure info
SCAMP	Scalable Group Membership overlay with gossip-based Forwarding
Scribe	Multicast used atop any KBR overlay
Spanning Tree	Tree overlay for fan-in routing
TBCP	Wide area multicast overlay

Range of network services in different environments



Generality (Extending the Pattern)

Pattern successful for 8 overlays and is extensible



Ease of Use

- >15 programmers with different levels of programming experience
 - Domains: pub-sub, sensors, etc.,
 - Roles: Users, plug-in developers, configurators
- Generally followed the pattern
 - Suitable for 3rd party deployment
- 2-8 weeks development time
 - Based on complexity
- Need for clearer documentation and transparency of configuration techniques



Configurability

Profile	No. plug-ins	No. config. rules	Disk mem. for config. rules (KB)	Disk mem for plug-ins (KB)	Total No. of configs available
Empty	0	0	0	60	1
WSN	7	6	16	146	4
Multicast	21	19	59	169	89
Full	40	31	87	252	26,999



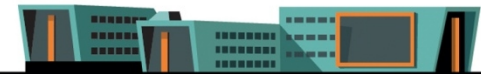
Resource Overhead

Configuration Name	#plug-ins	#Conns	Profile	Config. time (ms)	Dynamic mem. (KB)
Empty	0	0	Full	N/A	10,448
Empty	0	0	Sensor	N/A	8,352
Spanning tree	5	12	Sensor	191	11,452
Spanning tree	5	12	Full	193	15,264
TBCP	6	12	Full	211	15,144
SCAMP	5	9	Full	152	13,708
Scribe/KBR	9	27	Full	486	16,652
Scribe + TBCP	13	39	Full	592	16,972
TBCP+SCAMP	10	21	Full	281	15,308



Comparison to Key Related Work

- Micro-protocol frameworks
 - Ensemble, JGroups, Cactus, Appia
 - Open Overlays tackles wider heterogeneity, and offers greater software architecture support for configuration and reconfiguration
- Overlay Engineering Frameworks
 - P2, Macedon, OverlayWeaver, etc...
 - Simplify development, but no subsequent support for configuration, reconfiguration...



Concluding Remarks

- Presented and evaluated Open Overlays
 - A framework to address network heterogeneity
 - Usefulness of virtualisation
 - Configuration of combined overlays
 - Reconfiguration management
- Questions ?

