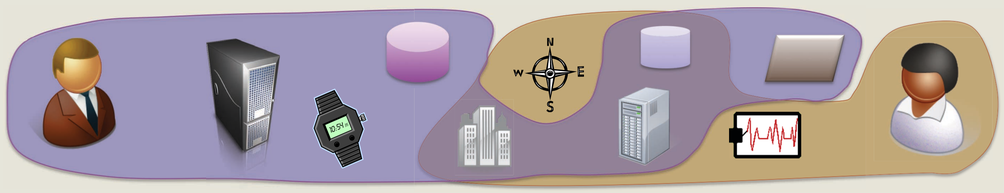
Policy-enforcing middleware for emerging systems

Computing is becoming increasingly ubiquitous. The number and range of system *components* (data producers/consumers), many of which are dynamic and/or mobile, is dramatically growing.

To get the most from emerging systems, components must not operate solely within vertical silos (a single application/system). Instead, components should be free to be used and reused, interacting with any other, whenever appropriate.



Users and environments will coordinate components in a number of different ways, to achieve a range of functional goals.

Policy encapsulates high-level (user) concerns/preferences. It is enforced by undertaking reconfiguration operations, in the relevant circumstances, to meet high-level goals.

Action	Component	Parameters	Description
connect	phone.map	local, owner:Tescbury's, data:location	Connect the phone's mapping app to any component that a) serves interpretable location data, b) is on the local (shopping centre) network, c) is owned by Tescbury's
change_privilege	phone.adverts	allow, from:Tescbury's	Allow the phone to receive advertisements from Tescbury's
change_privilege	phone.adverts	deny, from:McRonald's	Do not allow advertisements from McRonald's
connect	home.fridge	local.trolleyservice, owner:Tescbury's	Initiate interactions between the smart fridge at home to the smart shopping trolley in the store, to help with shopping lists, etc.

Policy for a user entering a shopping centre, enforced by their phone.
Mobile policy engines facilitate seamless interaction with different environments.

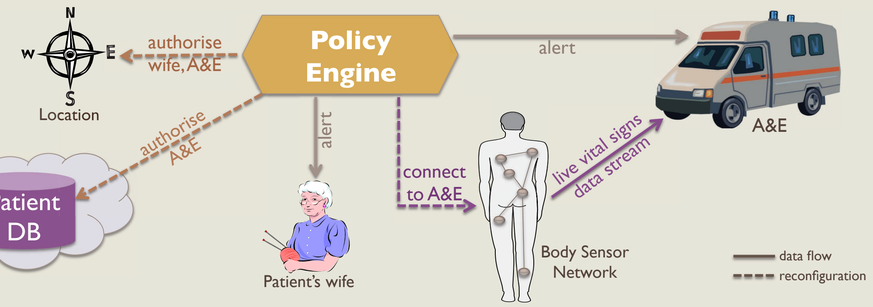
Policy engines are components dedicated to effecting the preferences and functional goals of an individual or environment.

SBUS is a messaging-middleware we developed for mediating between components and networks, to manage their interactions.

SBUS allows dynamic reconfiguration, to control:

- Connections: when and where to connect/disconnect; how to interact (RPC, streams, publish/subscribe);
- Translation: how to interpret the data (and types) of others;
- Security/Privacy: permissions, encryption, discovery, filters.

The reconfiguration operations allow components to themselves, *or as instructed by others*, adapt as required.



Reconfiguration in a medical emergency. Patient-specified policy enables a proper care response by *automatically* relaxing restrictions, and informing the relevant parties.

This approach aims towards the pervasive vision, as it:

- *Enables wide-ranging **functionality**:* components may be used/reused for a range of purposes, independent of component logic.
- *Facilitates **customisation**:* to individuals and environments. Users rather than developers specify how/when services are used.
- *Gives **control**:* allows fine-grained specification of what data may be obtained/shared, and under what circumstances.
- *Supports **ad-hoc** interactions*: rather than coordinating through a distant central server. Useful for smart home/city scenarios.

Jatinder Singh – Computer Laboratory - University of Cambridge