

# **Autonomic Communication**

## **Research Agenda for a New Communication Paradigm**



**Fraunhofer** Institute for Open  
Communication Systems

## Table of Contents

<b>DEFINITION AND SCOPE.....</b>	<b>1</b>
<b>AUTOCOMM APPLICABILITY AND EXPECTATIONS .....</b>	<b>1</b>
<b>AUTONOMIC COMMUNICATION PRINCIPLES .....</b>	<b>2</b>
1. GROUP COMMUNICATION .....	3
2. EVOLVABILITY.....	4
3. COMPOSITE FUNCTIONAL SYSTEM .....	6
4. DISTRIBUTED POLICY-BASED CONTROL .....	7
5. PROGRAMMABLE RULE-BASED SYSTEMS.....	8
6. DESIGN FOR CONFLICTS .....	9
7. ROLES IN THE MIDCOM.....	10
8. CONTEXT AWARENESS .....	12
9. SELF-SIMILARITY OF SELFWARE .....	14
10. SECURITY: AUDITING AND DANGER MODELS .....	16
11. AUTONOMIC ROUTING.....	17
12. COMPOSITION .....	18

## About this document

This document outlines a draft research agenda for a new communication paradigm dubbed Autonomic Communication (AutoComm). The purpose of this publication is to request comments from relevant R&D communities, and based on this to propose a long-term research within future and emerging technologies area. Comments and suggestions should be sent to the mailing list: [compar2020@fokus.fraunhofer.de](mailto:compar2020@fokus.fraunhofer.de), or directly to the author, Michael Smirnov: [smirnov@fokus.fraunhofer.de](mailto:smirnov@fokus.fraunhofer.de)

## Definition and Scope

Autonomic Communication (AutoComm) research studies the individual network element as it is affected by and affects other elements and the often numerous groups to which it belongs as well as network in general. AC's goals are to understand how desired element's behaviours are learned, influenced or changed, and how, in turn, these affect other elements, groups and network.

The following tasks performed by a network element are pertaining to the AutoComm.

- Learning by network element accomplished through sensing and perceiving and occurring at every moment of its existence;
- Participating in group communication, i.e. in communication within purposeful (structured and unstructured, ad hoc) communities of other elements brought up to existence by network element's being a component of a composite functional system (CFS),
- Understanding through interpretation and evaluation of perceived information and knowing the meaning.

## AutoComm Applicability and Expectations

Autonomic Communication enables meaningful, and/or purposeful communication, it follows the idea of driving communication from the context in which it is being used.

- In **human-to-human** telecommunication AutoComm will empower network elements transparently to humans to self-organise to best fit the needs of a session, they will learn user intentions, they will directly observe and react to context events in the network and in the real world, without explicit user interaction;
- In **business-to-business** communication the promise of AutoComm is to allow communication and cooperation that are more seamlessly integrated into business tasks and environments;
- In **human-to-machine** communication that will happen everywhere when computers are built into artefacts by monitoring the events that happen around them, an AutoComm system will be performing services autonomously, or will be adjusting the behaviour of services the user might request implicitly or explicitly;
- In **machine-to-machine** communication AutoComm will

help components to self-organise into composite entities, optimally providing required, often complex functions.

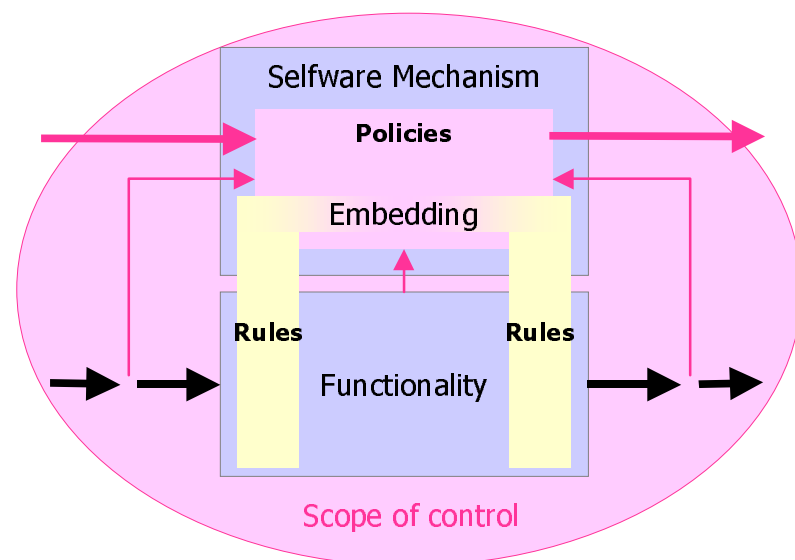
Ultimate contribution of Autonomic Communication research and development will be to enable an evolving communication technology platform for sensing, communicating, decision making and reacting, ready to serve in networking and networked businesses. AutoComm's fitness to **Critical Mission Systems** and to **non-conventional networking** seem to be very natural and is to be covered in a separate document.

## Autonomic Communication Principles

**Selfware harmonizes concerns of local, autonomic control and global self-organisation to provide better services**

Autonomic communication is centred around **selfware** – an innovative approach to perform known and emerging tasks of network control plane, both end-to-end and middle box communication based. Selfware assures evolvability, however it requires generic network instrumentation. Figure 1 outlines a generic framework of a network element that is enhanced by a selfware mechanism to exchange generic policies with groups of other elements and, through embedding of policies to functionality rules that control the behaviour of an element.

**Selfware** mechanism is used to exchange generic policies (rules, defining choices in the behaviour of functionality) within relevant groups; those policies are made conflict-free, safe and secure through **embedding** (universal fine-grained multiplexing of generic policies) into functionality as its native rules.



**Fig. 1** Autonomic Communication element

Selfware principles and technologies borrow largely from well established research on distributed systems, fault-tolerance, etc., from emerging research on non-conventional networking (multihop ad hoc, sensor, peer-to-peer, group

communication, etc.), and from similar initiatives, like Autonomic Computing of IBM, XG of DARPA, Harmonious Computing of Hitachi, Resonant Networking of NTT, etc.

The rest of the paper is an attempt to draft major AutoComm principles, that might need revision and critical assessment in order to put them at work. Subsequently, some items of AutoComm research agenda are drafted along with principles. Clearly, a high-level research is needed that will define priorities and interdependencies between the below principles.

## 1. Group Communication

---

A network element is inherently a member of multiple groups, defined by physical and logical adjacency, by control and management functions, by participation in service provisioning, and thus serving an instant group of applications, by sharing of common configuration data, by relying on common power supply, etc. Often, however a network element is not aware of its group membership and can not therefor make any use of it. The following list of '7S' highlights generic group communication benefits:

- Simplicity – treat a system as a single unit
- Safety – protection of sub-systems
- Specialization – operation for *all* or for *several*
- Structure – encapsulation defines boundaries
- Substitution - an alternative implementation is easy
- Self-Reflection - group state information
- Self-Organisation - group infrastructure logic

Over years several flavours of group communication<sup>1</sup> have been implemented to meet particular application needs; for instance '*announce and listen*' is useful to establish a group of entities that wish to collaborate within a given session; "*publish and subscribe*" is good for information distribution, while "*claim - collide*" is probably the best for negotiation of common resource usage in very large and loose groups. Autonomic Communication offers yet another flavour that is dubbed "*entity-group*". An entity-group is programmable and controlled group communication; *leaves* and *joins* to a group are performed by an element based on *membership rules* that comprise part of network element's

---

**Group communication for control plane tasks: programmable and controlled**

---

<sup>1</sup> Not to mention most generic a native IP multicast flavour of group communication

behaviour and are distributed over a control channel. How an entity will act within a group will depend on entity's *group behaviour definition*.

A targeted research in group communication within AutoComm projects will examine the problem space in inter-relation with other principles and paradigms. Of particular interest are the following problems.

1. Optimal trade-off between group infrastructure overhead and particular network element function, taking into account various implementation paradigms of group communication, such as broadcast, multicast, anycast, and epidemic communication (aka gossip);
2. Adaptive group communication that while being sensitive to the varying communication context will adapt the group behaviour patterns accordingly;
3. Scalable infrastructure-less programmable and controlled group communication;
4. Scalable self-management of two basic types of groups: homogeneous (e.g. for load-balancing), and heterogeneous (e.g. for network service creation);
5. Naming, addressing and identities architecture for evolvable group communication.

## 2. Evolvability

---

*“Evolvability - designing to be part of something else”*  
*[Tim Berners-Lee].*

This phrase highlights two issues in design for evolvability: design of yet unknown and design to be a component of this yet unknown. Evolvability adds one but important dimension to the scalability feature of a system. We say that a system scales if it can easily be modified to fit the problem area, usually - demand. Evolvable systems modify themselves, so that they can be termed *self-scalable*.

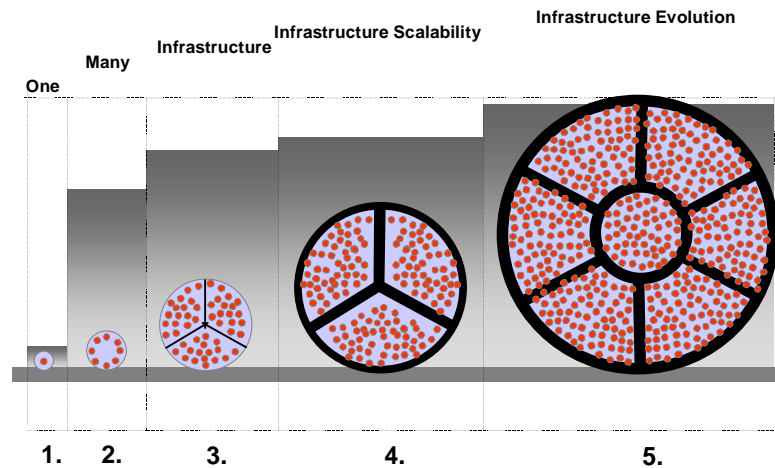
Let us demonstrate this by an abstract example, in which we consider five steps of a system growth (Fig.2). A system (circle) is intended to handle requests (dots). Assume our system scales easily from handling just one request to several. However, starting with the step three, a major change in the system design is needed, namely grouping of requests and subsequent partitioning of system resources, i.e. we discovered the need for an *infrastructure*. We also assume that further growth (step four) is relatively easy to accommodate by adjusting the same infrastructure. Yet further, at step five infrastructure itself requires radical

---

**AutoComm targets self-scalable (evolvable) systems; this will require (otherwise redundant) selfware functionality**

change. Thus, scalability demanded four major interventions, at each step the two types of changes were needed: adjustment of infrastructure (of internal functionality), and adding more resources (storage, processing, etc.) to the system.

Scalable system may require adjustments in both resources and functionality while it grows. An evolvable, *self-scalable* system needs only resource adjustment - functionality adjustments are made via self-organisation, though a functionality for doing this has to be provided at system's 'birth'.



**Fig. 2** An example of scalability

A sketch of the same growth enabled by an evolving system might utilise the *separation of concerns* between resource and functionality adjustments. This separation is fundamental, because adding resources is trivial but can't be made by a system itself (it has at least to be authorised for doing this), while functionality can be self-adjusted. Note, however that functionality facilitating evolvability has to be present within a system in certain form, which might seem as redundant if evolvability is not taken into account. The challenge of evolvability design is to develop successful *behaviour patterns* and *rules* of their applicability; these are to be coupled with self-awareness (self-sensing) feature of a system.

Research agenda in the area of evolvable systems might include the following topics.

1. Understanding mechanisms and boundaries of controlled complexity and controlled flexibility of infrastructure adjustments;
2. Understanding patterns of successful evolution behaviour and associated thresholds;
3. Developing a formal framework able to capture evolvability patterns in autonomic communication, specify them in behaviour definitions (BD) and offer a set of operations for

- manipulating BDs and their components;
4. Methods for adaptive composition of behaviour definitions and rule-based systems for their access control and self-management;
  5. Understanding of catastrophic states in evolvability design space and design of suitable reaction patterns;
  6. Security of evolvable systems, basically research in evolvable security systems, in particular role-based access control, evolution of rule based systems and the like.

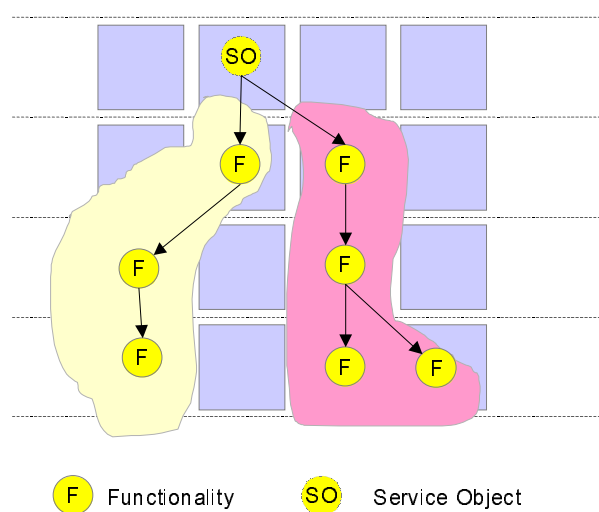
### 3. Composite Functional System

---

Understanding of control plane complexity is critical; this complexity is ever growing. Management and security are essential ingredients within existing functions of network control plane; AutoComm adds to it a desire to support yet unknown Internet service architectures.

Autonomic Communication offers to revise the very notion of a function by departing from a strict binding of components delivering a function and replacing it by on-demand composition of components; this is a concept of a *Composite Functional System*. Strictly, the CFS is characterised by a presence of *a constant (invariant) task, performed by variable (variative) mechanisms, bringing the process to a constant (invariant) result that is characterised not only by complexity of its structure but also by mobility of its component parts.*

A composite functional system defines its invariant task as a *service object*, that itself has zero *functionality* but is used to control and configure multiple bundles of functionality that might be used to provision a service



**Fig.3** CFS in Layered Architecture

Control plane issues are to be considered as cross-layer issues that are spanning down to hardware e.g. for the support of signalling and QoS, and upward to overlays e.g. for the support of interaction of overlay routing and network layer routing.

The R&D agenda of CFS in Autonomic Communication has two large areas.

1. CFS research to solve *traditional* tasks: robustness, complexity of control, adaptability, modes of failures, safety, predictability, manageability, evolvability, and security;
2. CFS research for *innovative* tasks, such as CFS reconfigurability, role-based addressing of CFS components; self-organisation; on-demand coupling of different composite functional systems that will allow true *cooperative networking* - seamless inter-working of networks with different technologies and business models.

#### **4. Distributed policy-based control**

---

*“Policy is a rule defining a choice in the behaviour of a system” [Moris Sloman].*

As such, a policy system is not a functionality, but a ruleset to control and configure a functionality. Existing use of policy is rudimentary for the reason that a functionality to be controlled by a policy was typically not designed to accept its behaviour choices through policy control interface. However there is a growing demand to implement policy control in networking (examples are DARPA neXt Generation Communication programme, IETF and DMTF work on policy, Resonant Network Architecture of NTT, Harmonious Computing of Hitachi, Autonomic Computing of IBM, related work at W3C, etc.). The interest in policy based control is high because it can be made inherently safe and secure; it offers much higher flexibility for operators.

---

**Policy in AutoComm will self-organize via policy computation and distributed conflict resolution and policy context**

Autonomic Communication goes beyond this and proposes a proactive approach, in which policy system will self-organise. AutoComm proposes to close a control loop within policy framework by extending it to sensing, perceiving and reaction. AutoComm’s policy framework is not limited to policy decision and policy enforcement logical points; we propose to include at least *policy computation* (where multiple entities could possibly inject components of would-be policy, and where conflict resolution will take part) and *policy context* (where additional information e.g. for conflict resolution will be derived from) logical points. Thus,

AutoComm's policy framework will directly address the issue of design for tussles in the cyberspace [NewArch].

The following research directions are proposed.

1. Dynamic multi-party conflict free policy computation algorithms;
2. Context driven conflict resolution in policy computation
3. Understanding policy context; formal methods for context definition and handling;
4. Multi-aspect conflict-free policy computation (e.g. security policy and QoS policy, etc.),
5. Self-organising ontologies for policy interworking and interoperability, etc.

## 5. Programmable rule-based systems

---

*“Rule-based Systems Security: Designing security policies and a framework for policy management that will allow necessary access to systems and data in previously unplanned ways, and by persons and systems not normally permitted to do so, is a big need”*

*[Report from the NSF workshop Responding to the Unexpected, <http://crue.isi.edu/research/report.html>].*

There is no universal understanding of access control, though the overall concern is flexibility. Hence, ever increasing attention is paid to role-based access control models and algorithms. However existing work including recently proposed NIST standard for RBAC<sup>2</sup> does not allow unexpected access to a resource; it is generally referred to as an unresolvable conflict. Autonomic Communication proposes an approach separating the concern of access from the concern of its programming. Hence, two types of access control rules are considered - embedded and programmable. Embedded rules are optimised for request handling, while programmable rules are optimised for decision making, including conflict resolution. Embedded rules are proof-based conflict free at any instance of time, and the process of embedding is logged after every change in such a way that *auditing* and *reverse engineering* of a rule base are always possible and are standard for access to any network functionality.

The following research topics are to be accomplished:

1. Formal representation of a rule-based system security model for

---

<sup>2</sup> RBAC – Role-based Access Control

---

**Programmable rule-based systems can solve *unexpected* requests; dynamically synchronised logging allows complete history reconstruction**

potential standardisation of its representation and a set of operations;

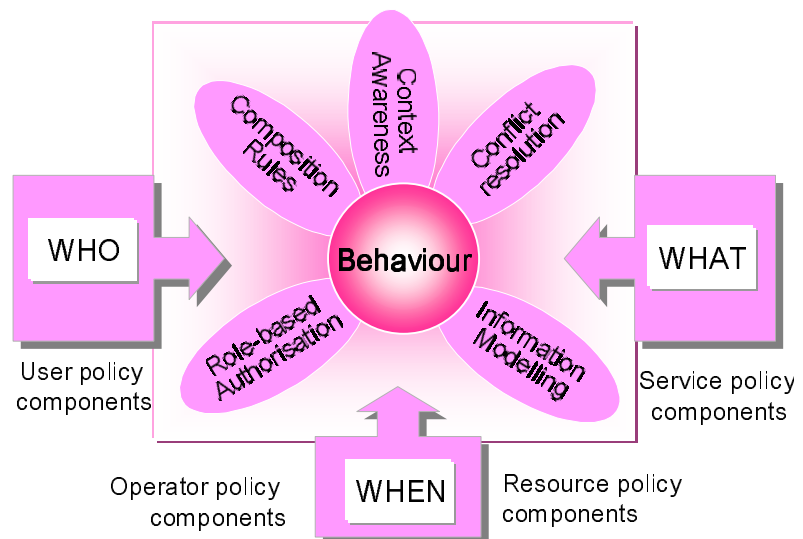
2. Development of efficient embedding algorithms suitable for implementation in hardware and software;
3. Research in conflict resolution theory suitable for RBAC and programmable RBAC;
4. Fitting of the above into distributed policy-based control framework of AutoComm.

## 6. Design for Conflicts

Conflict is a feature. Cyberspace is full of tussles, which we have to be prepared to design for; the *NewArch* project<sup>3</sup> at ICIR suggests the following design principles:

- Design for variation in outcome, so that the outcome can be different in different places, and the tussle takes place within the design, not by distorting or violating it. Do not design so as to dictate the outcome. Rigid designs will be broken; designs that permit variation will exist under pressure and survive technical;
- Modularize the design along tussle boundaries, so that one tussle does not spill over and distort unrelated issues, and
- Design for choice, to permit the different players to express their preferences.

Autonomic Communication welcomes multiple sources of policies defining choices in the behaviour of network elements, autonomic policies injected by parties are subject to composition, conflict resolution, and, when needed to enhancement with the help of contextual information – all under strict authorisation and conformance checks.



**Fig.4** Desired behaviour as conflict resolution

Clearly, the design for variation in outcomes in Autonomic Communication is the major concern of *Composite Functional Systems*, while definition of tussles and

<sup>3</sup> <http://www.isi.edu/newarch/>

tussle boundaries is a communication service issue. Usually existence of a conflict is recognised when the design is implemented; thus *design for conflicts* has to take unusual perspective on a problem area. Practically, this might require to revise many commonplace assumptions on trust, priorities, services, connectivity, and the like. Finally, to permit the different players to express their preferences we need, again to change our understanding of communication setting up fundamentally; a simple user service request is no longer to be considered as pertaining to a specific *dedicated* protocol only. We have to develop methods that would allow networks to treat such a request as expression of user preferences, as a micro-SLA, finally as a rule - an *atomic policy*.

---

**Conflict resolution in  
Autonomic  
Communication is instant  
and autonomous**

With this understanding we can enrich semantics of already deployed protocols, allowing negotiation, adjustment, harmonisation of requests by mechanisms of conflict resolution applied to rules. Computational conflicts have been studied in AI, in multi-agent systems and the like. There is a significant difference in requirements for AutoComm conflict resolution in comparison to these related works: conflict resolution in autonomic communication is *instant* and *autonomous*, meaning that conflict resolution functionality and information has to be readily available at the place and at the time of conflict detection.

The following research topics are relevant:

1. There is a need to develop theory of conflicts suitable for Autonomic Communication; on one hand side this is a tough task because conflicts reflected at the AC level are rooted in the human space; on the other side, the task is feasible due to its constraint nature. Theory of conflicts might be developed in several flavours, applicable for various facets of AutoComm design, such as CFS, group communication policy, rule-based systems, etc.
2. A formal framework for semantics enhancements of conventional protocols should be developed that should become a part of a *design for conflicts* methodology.
3. A methodology of composition of network policies out of atomic policies has to be developed and validated (cost/efficiency, etc.)

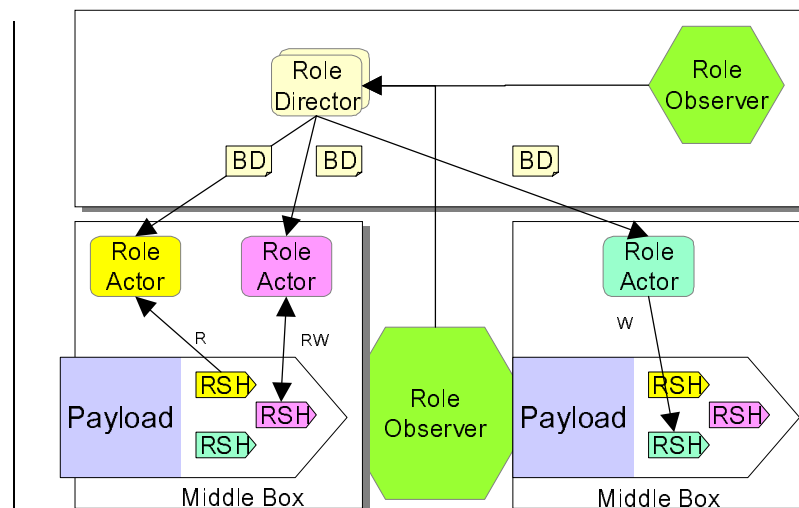
---

## **7. Roles in the Midcom**

Non-conventional approaches to networking differ significantly in what is taken out of convention. Recently proposed as part of NewArch project role-based architecture

(RBA) copes with ever growing complexity of multi-service Internet in a radically non-conventional attempt to substitute protocol layers by a new abstraction - a role, *functional specification of a communication building block*. Major motivation for RBA is that traditional layering is de-facto violated by many Internet protocols (e.g. mobility support in IPv6 keeps host route table at layer three; wireless hosts with multiple radio interfaces are injecting layer three to layer two commands from layer four, etc.), as well as of sub-layers (e.g. MPLS is actually layer 2.5; IPSec is layer 3.5, and transport layer security is 4.5, etc.). Giving up layers also means giving up *layer secrets* - particular ways of layer function implementations and clear exposure of roles that are assumed to be more open. A role can be addressed directly, without encapsulation into layer specific protocol or into interface data unit. Roles can interact much more flexibly than entities hidden in layers and obliged to follow layer conventions.

Role-based architecture [NewArch] is extended with meta-roles: **role director** to upload modified behaviour definitions (BD) to role actors, and **role observers** – to sense the network conditions and to trigger role directors



**Fig. 5** Roles and meta-roles in Role-based Architecture

Giving up layers is tremendously difficult - layering is the most natural way of thinking about complex systems, because layers are successfully separating concerns of what is to be each layer's function. Fortunately, layering is not the only paradigm for separation of concerns inside a role actor. Autonomic Communications proposes to use policy-based approach for the same separation of concerns. This will be made feasible by (see *distributed policy based control*) means of behaviour definitions and conflict resolution. It is envisaged that a number of *meta-roles* associated with role's actor might become needed, such as role director and role observer. Role actors become active when triggered by role specific headers found in IP datagrams. Role directors are triggered by a need

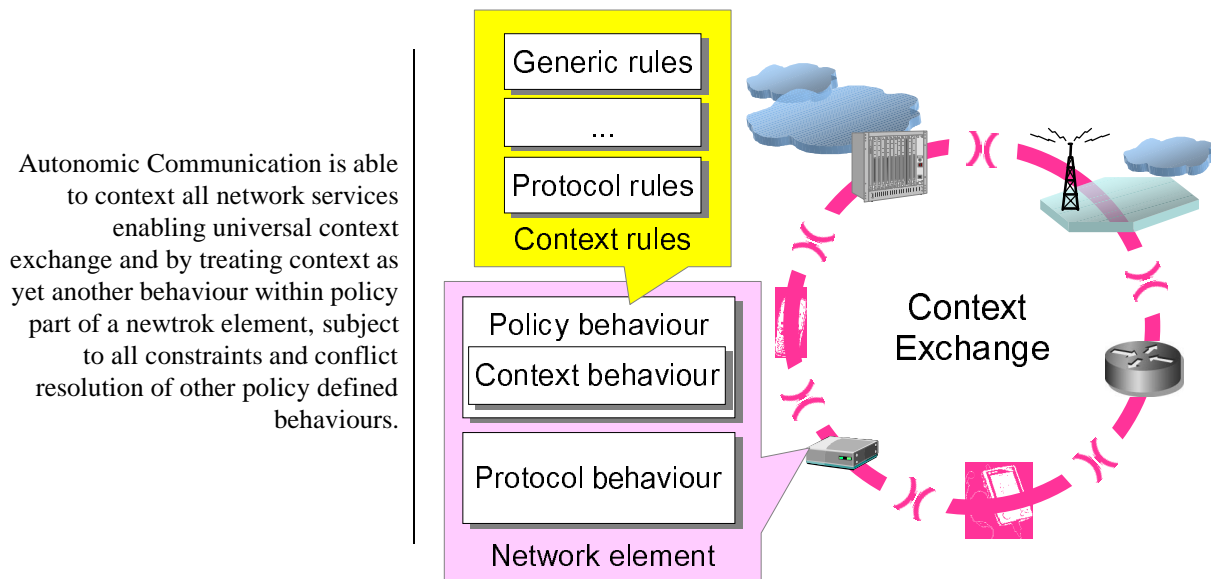
to specify/activate/enforce new behaviour at associated role actors; role observers are triggered (as specified in their behaviours) by any of the above events pertaining to the task of monitoring or auditing of behaviours of role actors and directors.

## 8. Context Awareness

*Context, v. To knit or bind together; to unite closely*  
[Webster's 1913 Dictionary]

*The whole world's frame, which is contexted only by commerce and contracts. [R. Junius]*

Autonomic Communication is focussing on behaviour of a network element in juxtaposition to numerous groups it relates to. This group-oriented approach opens yet unknown opportunities to assist communication with a context information (personal, location/movement, and media/ session related). Because *one man's context is another man's text* it seems obvious that group communication will enable richer information sharing between group members. However, from traditional protocol design viewpoint a group member *making use* of a shared information uses it as 'text' hardwired in the member's behaviour defined by the protocol<sup>4</sup>.



**Fig.6** Enabling context-driven communication

Autonomic Communication departs from this viewpoint by suggesting another definition of a behaviour, namely the

<sup>4</sup> Without lack of generality we consider a network element implementing just one protocol, that is having just *one function* to contribute to Autonomic Communication of groups of elements; generalisation to multiple functions is a straightforward one.

one that immediately follows from the definition of policy by M. Sloman - policy is a rule defining choices in the behaviour of a system. In this view policy and behaviour are two separate concerns that can be engineered separately and independently. We shall call the latter a *protocol behaviour*, while inter-working of protocol behaviour and policies will exhibit *element's behaviour*. Further, because policy is a rule 'IF *condition* THEN *action*' it is easy to define context aware rules, i.e. ones where conditions will represent contexts that are desired to be taken into account. Thus without altering the functionality protocol (i.e. without reboots or even suspending element's operation) new policies can change element's behaviour triggered by changes within element's context.

Implementation of a network element following the above design will have *at least* two parts communicating through an internal interface - a policy behaviour part, and a protocol behaviour part, somewhat similar to protocol MIB and a protocol itself within current SNMP based management. Unlike SNMP, Autonomic Communication offers evolvability by allowing hierarchy and recursion.

A *hierarchical context* awareness means that a policy behaviour can be organised as nested rule-sets, from generic rules down to protocol specific rules. Generic rules form the outmost ruleset, these rules enable Autonomic Group Communication of an element, they are used by network element to discover, to join/leave, to monitor, etc. all groups an element has relations to. Other rulesets have more specific relation to element's protocol, i.e. they contain *rules of context exchange* with more (for outer rulesets) or less (for inner rulesets) groups to which the element participates.

A *recursive context* awareness in Autonomic Communication means that network elements by participating in context exchange trigger each other to perform their functions, thus being *mutually recursive*. This happens because context in Autonomic Communication is treated as yet another behaviour. The purpose of all behaviours (loosely referred to as *selfware*) found in the policy part of an element is to assist self-organisation of network elements within a group and between groups. The latter requires *context propagation rules* to be deployed by an element participating to more than one group.

## 9. Self-similarity of selfware

---

Nested rulesets (see *context awareness*) are comprising a distributed selfware that facilitates self-organisation of network elements with the purpose to enhance, optimise, configure, etc, their protocol behaviours. Rulesets are exchanging events<sup>5</sup>, events carry changes to the context. When event is received by an element whose policy part contains a rule that is triggered by the event then rule's action is performed. This action can be an invocation of a protocol function locally or remotely, or a sending of another message, each event is subject to authentication. Thus, each policy ruleset in Autonomic Communication exhibits a behaviour that consists of arbitrary sequences of the following four types of actions:

*<admit, invoke, request, notify>*,

where *admit* performs *event authorisation (EA)*, i.e. it drops all events a ruleset is not subscribed to, and authenticates sources of those that are subscribed to; *invoke* is performing protocol *behaviour authorisation (BA)*, i.e. an access control to element's protocol functionality; a *request* type of action implements *behaviour obligation (BO)*, i.e. it sends an event to trigger explicitly invocation of certain function from another group member; finally, *notify* type of action implements *notification obligation (NO)*, i.e. it issues an event signalling the result of the network element function invocation. All types of actions are represented by rules, or by clusters of rules.

---

<sup>5</sup> 5An event in Autonomic Communication is  $\langle A, B, C, D \rangle$ ; it is defined through an action (*A*) that happened at network element - a box (*B*) that produced a set of post conditions (*C*) that are considered to be valid within duration *D*; both *A* and *B* are needed for event authentication; they bear *identity of event source*.

Selfware structure in AutoComm is self-similar in a sense that any selfware behaviour is structured as intersection of *authorisation* and *obligation*, and, from another viewpoint – as intersection of *eventing* and *behaviour*.

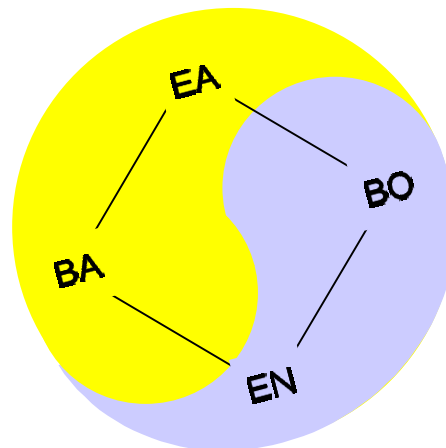


Fig. 7 Self-similarity of rulesets in selfware

The above structuring of rules to support ruleset action types demonstrates several levels of separated concerns inside<sup>6</sup> a ruleset. First, authorisation (*EA*, *BA*) is separated from obligation (*BO*, *NO*); at the same level eventing (*EA*, *NO*) is separated from behaviour (*BA*, *BO*). Further, within authorisation, we separate authorisation of requests (*EA*) from authorisation of requested action invocations (*BA*); explicit request for an action (*BO*) is separated from an implicit (*NO*) one. This creates opportunities for very efficient (fine grained) *security and safety* of Autonomic Communication systems. At the same time this offers universal interface between AutoComm systems and between system components. We call this feature external self-similarity of a ruleset; it is possible because of internal self-similarity: all four rulesets are self-similar in a sense that there can be no dependencies between rule sets with regard to access rights for the same identity in different rule bases. For example, event authorisation rules may deny events within certain sub-ranges<sup>7</sup> of source identity, however admitted events sourced by allowed identities may still carry denied identity values in their payloads. Thus, *EA* rules having the entire ID range as their input range may shrink it as their output range, but it is expanded again to the entire ID range for behaviour rules and/or notification rules.

<sup>6</sup> Recall, that a ruleset itself is a result of separation of behaviour and policy concerns

<sup>7</sup> We assume that identities of network elements are assigned from the same common and flat range of identifiers, ID.

## 10. Security: Auditing and Danger Models

---

*"The ... cell, awakened by the ... alarm signals then starts the chain of events that sets an immune response in motion."*

*[Matzinger, P. A]*

---

**AutoComm's immune system is embedded into selfware and is controlled by auditing rules and danger models discovery and propagation rules**

Autonomic Communication will rely on its own immune system to assure security and safety, to detect early and to isolate intrusion and DoS attempts on selfware. Autonomic Communication assumes that *immunity has to be embedded into selfware functionality*. Additionally to whatever security measures are available at the level of element's protocol a selfware adds value to it. Like everything within selfware, AutoComm's immune system is implemented by security rules. Security rules are derived from security policies, when available and from selfware behaviour definitions. Embedding of security rules happens concurrently with embedding of selfware rules, thus it is not possible to modify selfware's behaviour without proper modification of associated security rules. In fact, adding of derived security rules is *a part* of the embedding process along with conflict resolution. There are potentially two sources of control information for the embedding. First, features of the function, and, second features of a distributed service being created/modified by changes in selfware behaviour.

In their embedded instance AutoComm security rules appear in two forms - auditing rules and danger models discovery and propagation rules. *Auditing rules* are observing from inside how selfware's behavior is being accomplished and are relating the progress to functional invariants (see CFS), such as timers and thresholds. The functional invariants altogether constitute *interlacing bounds* of safe and secure behaviour; while the behaviour stays within these bounds the auditing system is observing it silently. *Danger model discovery and propagation rules* are being triggered by auditing events generated whenever selfware's behaviour is not within the bounds. Danger model is a forbidden behaviour pattern, it is represented as a set of rules with various violations associated to them. A cascade of actions can be triggered by auditing events, starting with a local cache lookup, and finishing with creating a proactive defense behaviour, where existing or on purpose created group of selfware agents will isolate a danger and minimize the damage.

## 11. Autonomic Routing

---

---

**AutoComm' s routing is self-organisation of groups of selfware units that collaborate on behalf of SSR entities and jointly modify actions performed by conventional routers**

Multiple conflicts are hidden within routing in the Internet, that could be classified into native routing conflicts and service specific routing conflicts. Native routing conflicts arise from name to address bundling conflict; host's name is usually what we seek, a name is mapped to an address either by applications (e.g. P2P) or by a name service resolver within IP module, and finally a name is mapped to a route by a router or a link layer. Service specific routing happens in large clusters of storage and computing devices, within GRID infrastructures and within application level middleware; the latter is also known as RMI- routing. The main requirement of service specific routing is to provide a *Transactional Message Bus* between service entities with needed topology, duration, QoS and other characteristics that are generally subject to a dynamic SLA. Service specific network can be seen as an overlay network over a public Internet (or Core Net) that has conflicting with SSR set of requirements. The core net has to be *fast, stupid, resilient and scalable*.

Autonomic Communication proposes a selfware routing infrastructure that mediates between SSR and Core Net. The selfware routing has to be both scalable - to cope with the Core Net, and programmable - to meet the SSR demands. Major research efforts are foreseen at this frontier: there is a need for a new paradigm (How the routed data are seen at this level?) It has to be a view that combines transactions, flows and datagrams. There is a need for new addressing (What are the entities acting within selfware routing? How they are addressed? How they communicate?), for new mechanisms, and the like.

A unit of selfware routing has to be able to reflect, differentiate and to maintain its multiple relations to underlay (Core Net), to overlay (SSN), and to other units of selfware. Selfware routing is not yet another layer, it's rather a parallel world co-existing with conventional (Core Net) and emerging (SSR) routing. Autonomic Communication sees this parallel world as self-organising set of groups of units that collaborate on behalf of SSR entities and jointly modify actions performed by conventional routers. Thus, conventional routers can concentrate routinely on forwarding leaving most of routing control exchange to selfware routing.

Here is a *sketch of potential concept*. A Core Net router maintains a forwarding table that for the purpose of AutoComm is convenient to represent as a set of simple rules, where datagram header presents a set

of conditions, while forwarding to a particular outgoing interface is the only action, except filtering of martian addresses and the like. Nowadays, forwarding tables have to store values for all possible combinations of header's data; this simplifies forwarding decision making because a router has just to make a FIB lookup that returns a decision. The simplicity of forwarding has a great burden on routing: it has to propagate to every router all changes in topology, QoS policies, transit policies, and the like; all these contributes to forwarding tables growth especially at the level of BGP. The above propagation is done by advertising of address prefixes reachable through advertisers; to meet requirements of multiple policies (resilience, QoS, transit, etc.) advertisers chop a single address prefix reachable through them into multiple sub-prefixes characterised by different levels of their ability to meet the above policies. It is always possible to define a [quasi-] constant part (maybe just default routes) in any FIB, with constant part updates being really rare events (e.g. with major topology changes). Thus FIB consists of two largely different from the update frequency parts - [quasi-] permanent, and variable. Further, in variable part we can distinguish between several *contexts* - transit (peerings of AS), QoS, multi-homing (resilience), and the like. Actually, update of a forwarding database is a process of re-writing of one set of forwarding rules by another one. This process is controlled by a routing manager - based on all routing updates received from its neighbours and based on decision making rules this manager selects from alternatives for forwarding, thus making forwarding unambiguous.

---

**In AutoComm different routing update contexts are treated as separate concerns**

The focus of AutoComm's selfware routing is on decision making rules. In selfware different routing update contexts are treated as separate concerns, thus updates caused by changes in domain's transit policies will be independent from updates caused by QoS or resilience policies. Selfware doesn't preclude these updates to come from different sources, they rather have to come from trusted and well-formed sources.

What is called here a selfware routing update context can also be expressed in terms of routing update roles. Independently performed updates (rule modifiers) are to be safely installed in a FIB. this process performed by an Selfware Installer (routing updates manager) with the help of rules that eliminate conflicts between original updates arrived from different sources and/or representing different contexts and between original updates and existing FIB.

---

**12. Composition**

---

Autonomic Communication enables composition of the 'functional systems', that always includes a series of *afferent* (adjusting) and *efferent* (effector) impulses. With any example of somewhat complex autonomic behaviour it proves to be impossible, in principle uncontrollable with only efferent impulses. Functionality of selfware separates concerns of authorisation of effector events from obligation to issue

adjusting events. To the level of embedded rules within a selfware these concerns are brought by a series of steps starting with understanding of what is a desired behaviour definition.

---

**In AutoComm composition of functional systems is achieved through their contention within collaborative zones**

Localization of components of a composite functional system cannot be limited to narrow zones (e.g. network edge or network control server) or to isolated groups of dedicated nodes, but it must be organised in systems of zones working *in contention*, each of which performs its role in complex functional system, and which may be located in completely different and often far distant areas of the network.

Collaboration of zones in contention may be based on certain external mechanisms (repositories, directories, etc.) that are essential elements in the establishment of functional connections between individual parts of the CFS, this can be called extrazone organisation of complex composite functions.

Relation between zones is never stable, especially during certain *learning* activities, such as discovery of each other's capabilities and contexts. The learning process results in fixing certain 'kinetic melody', starting from this point learned composite behaviour starts to depend on a different system of concertedly working zones, this is the beginning of *perceiving*.