

Network Self-Organisation Programming Language

Michael I. Smirnov, Fraunhofer FOKUS

Complexity is the Problem. The real danger of network complexity is the ever increasing human effort for network control and maintenance. Network operators attracted by seemingly easy to operate TCP/IP networks and their promise to serve as a really universal platform for a multi-service network now face the need to re-engineer TCP/IP networking to become *service-aware*. Service awareness means that not only all digital items pertaining to a service are delivered but also that all business relations pertaining to a service offer are fulfilled in the cyber world. Traditional Internet protocols are not really ready to be deployed under the control of business model logic, and IETF usually considers these business cases to be outside of its engineering scope, making exceptions for well understood service-awareness building blocks; best examples are AAA and policy. Probably, because service-awareness is natural in the world of connection-oriented communication, multiple attempts to re-engineer the Internet take a form of “Signalling System 7 over IP”. The failure of such a combination is predictable from the complexity, though orthogonal of both parts of the combination.

Self-Star is not sufficient. Network self-organisation along with self-management, self-healing, self-anything¹ is often regarded these days as a key enabler for more intelligent, more resilient and more performant network. It is always an assumption that self-organised network will *understand what is required* from it. This important assumption is often overlooked by network specialists and over-emphasised by e.g. AI and social communication specialists. We argue that both views are badly needed to meet human (users and providers) expectations. We propose a long-term research agenda inbetween currently disjoint viewpoints, that will require both types of researchers working along the same agenda and making a target-oriented effort to bring both viewpoints to a converging middle. In proposing this we firmly stay at the network side for a simple reason – deep understanding of network reality is critical for the pragmatic success of a long-term research initiative. The network needs both – understanding of what is required now, and ability to best self-organise itself to perform the task.

Network Self-Organisation as a Program. The vision of future network architecture can be captured as “Architecture is a program”², however this program will be an unusual one; it will run on a non-Turing computing *machine*³. This program will run simultaneously tens of billions of processes on billions of processors, these processes will pass intermediate results to each other based on logic of self- and service-awareness⁴, but shall stay ultimately autonomous during execution. Not only parts of this program will modify themselves but also the program will never be considered as completed. Traditional phases of analysis, planning, coding, debugging, etc. will not disappear

¹ Sometimes referred to as self-star (self-*)

² The slogan belongs to Michel Riguidel (ENST, Paris)

³ See e.g. B.J. MacLennan’s “Natural Computation and Non-Turing Models of Computation”

⁴ From now on we refer to service-awareness as a *task* of communication

however will run permanently. Thus, it will be absolutely legal to make coding during run-time. Moreover, we demand that programming is to be available not only for network operators but also for regular users, as well as for artefacts when considered eligible. There are a number of challenges of course. How should autonomous processes communicate to assure global purpose? Who defines the purpose? Does, for example a network provider's understanding of fairness help her to best satisfy user needs and to maximise its own revenue at the same time? How the network program shall understand its users when they push the magic 'Do What I Mean' button on their devices? How to guarantee that the network program will not become selfish, i.e. follow the Parkinson's law and use all of the resources for self-sustainability?

Direct Functional Programming. Self-organisation will best address the issue of *How* a task is to be performed by a network, while the issue of *What* the task is will be best addressed by the direct functional programming of a network by all network users. Network programming is essentially constrained by available functionality, by network owner policies, and/or by user community policies. We offer to revise the very notion of a function [of a network and network element] by departing from a strict binding of components delivering a function and replacing it by on-demand composition of components; with this view a network is 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 complexity⁵ 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 on-demand bundles of functionality that might be used to provision a service. 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 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, treated as danger models identification and propagation;
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.

⁵ This complexity might be higher than that of a system with elements binding per task, however it facilitates adaptation and evolution and the complexity stays constant over the evolution of a system