Propositions (Stellingen)

by Mihai-Lucian Cristea, author of

Parallel and Distributed Processing in High Speed Traffic Monitoring

1. The tree-like multiplexing of protocols and applications on a common link layer frame gives enough parallelism for a distributed processing environment. Both the OSI stack and specific layer properties (e.g., address or port numbers) provide classifiers which allow parallel and distributed processing of incoming traffic. [This thesis: Section 4.2 and Chapter 5].

2. Supporting heterogeneity in a distributed system allows for smooth upgrading of the system. Moreover, using heterogeneous processing units paves the way for efficiently auto-partitioning a main application into components supported by specialised units. [This thesis]

3. In a reconfigurable architecture, a key-issue to the tradeoffs between flexibility and performance is the synergy of control cores and specialised cores. The control cores give flexibility in building various applications while the specialised cores provide performance for certain applications. Moreover, a dynamic balance between control and specialised cores on request of the application would offer both performance and flexibility for the same reconfigurable architecture. [This thesis]

4. A network composed of network elements which support application specific components such as packet processing tasks and which are driven by the network data is a general purpose network. Such a general purpose network could support multiple service types (e.g., best-effort, end-to-end, mesh) coexisting on the same physical infrastructure made up of networked programmable network elements. Moreover, such a general purpose network can be extended with new services by deploying new software components into the network elements. [This thesis]

5. Binding applications to end-to-end lightpaths can be done safely and fast by enhancing the network data with specific tokens. The tokenised traffic would then choose the path when travelling across multiple network domains. Thus, the user application may enforce the required network demands through its transmitted data.

6. Defining a new language is simpler than building a compiler to support it. Compiler development is simpler than proposing an easy and useful language.

7. Using tools that translate the user requirements into hardware specifications is the first step towards self-programmable machines. Teaching machines the human experience in development and programming of new systems gives the freedom machines need to choose their evolution independently of human-beings.

8. Every computing machine built by human-beings has an origin by means of basic knowledge where it starts functioning from (e.g., BIOS for computers). However, a reconfigurable machine may loose its origin because of consequently (self-)applied updates. Therefore, such a machine needs a mechanism that always lets it return to its origin when there is a serious malfunction which cannot

be solved by applying updates. Moreover, to be effective, this mechanism must be an undiscoverable secret to the machine like the soul is to the human-beings.

9. An innovation often leads to an increase of the complexity of a system by adding new functions in order to break the current limitations. However, a radical invention would replace a complex system with a simpler and functional equivalent. Such an invention is a characteristic of the evolution of technology because the probability of failures in a system decreases with its simplicity.

10. Making mistakes is human, but to puzzle things irreversibly you need a computer.