Nimrod Design Goals and Principles

Technical Goals for Nimrod

Nimrod has an extensive list of goals, but this one tops the list:

- **Maximize the lifetime of the architecture**

This is by far and away the most important goal. To understand why, let's look at why large-scale communication networks - and the design philosophy that must be used in designing them - are inherently fundamentally different from all other large-scale software systems.

Design Philosophy for Large-Scale Communication Networks

- The design philosophy used for these systems is inherently different from that used in other large software systems.
- The need to *communicate* means that simultaneous use of alternative solutions is not really viable, as it is in all other software systems: e.g. operating systems, editors, languages, etc.
- This need to communicate means that if adding new capabilities means switching to a significantly different architecture, change will very difficult, as *upward compatability*, which is limiting, must *always* be observed.
- The sheer scale, and difficulty, of change in such systems means that major changes are very expensive and painful.
- Thus, to minimize costs over the *entire life-cycle* of an large-scale communication network architecture, maximizing the life-time of the architecture is the key goal.

Devolution

As *"fate-sharing"* was perhaps the key concept behind the last generation of networking architecture, the key concept of the next generation of networking architecture will probably be *"devolution"*.

Devolution, simply put, means that we will never do in the core of the network anything we can move to the edge. In other words, the network won't try and make decisions for the users, but rather will hand the responsibility over to the users.

This is related to the concept of "separating policy from mechanism".

The relocation of mechanism, from the network, into the hosts, is a key part of maximizing the lifetime of the architecture. The less the network does, the easier it will be to change and upgrade functionality.

Architectural Lifetimes and Engineering Viability

There is an inherent tension between producing an architecture with a useful lifetime (and how often have we seen crippling costs when an architecture runs into its limits), and producing an architecture which is viable, in engineering terms, at the time it is first deployed.

Overhead costs, etc, which seem trivial toward the end of the lifetime of an architecture (typically twenty years, or more) seem infeasible at the time the architecture is deployed.

However, without the boldness to pay high costs up front, to allow adoption of an architecture with more room for growth over the long run, down the line the architecture will be crippled by limitations imposed in order to produce engineering savings with only short utility.

There is no easy solution to this problem.

Additional Technical Goals for Nimrod

Nimrod has many other goals, which are ancillary to simply maximizing the lifetime:

- An incrementally/interoperably deployable system - without this, it cannot be deployed.
- Support *efficient* datagram mode service; i.e. the ability of a user to send packets without requiring any prior network setup and at low overhead.
- Scale indefinitely in terms of number of networks, routers, etc. The expected worldwide network system size is so large that optimizations based on a fixed maximum size aren't worth the limitations.
- Provide a very robust system, one which is proof against all kinds of failures *as well as* attacks.
- Provide flexible *firewalls* which allows the users to enforce as much security on the routing as they need.

Additional Technical Goals for Nimrod, Continued

- Support *"policy"* routing. "Policy" does not mean just restrictions on traffic; it includes all routing decisions (such as QOS) not made on a simple metric or simple combination of a few metrics. Policies can be divided into the following classes:
  - Restrictions on which traffic can use particular transmission resources (also called *transit policies*, or *access control*).
  - Restrictions on which resources can be used by particular traffic (also called *source policies* or the *trust model*).
  - Restrictions on who can find out the details about certain parts of the network (also called *information hiding*)

  An alternative term to "policy routing" which is used a lot now is *constraint-based routing.*

- Minimize the hand-configuration required of users, since this is a source of potential errors, as well as being tedious and complex.

Design Principles for Nimrod

- Do the best possible design, and then work out how to deploy it; don't simply imcrementally improve the existing system.
- Maximize the lifetime by:
  - Making the design as flexible as possible
  - Minimizing the mandatory, system wide part of the architecture
- Maximize robustness by:
  - Reducing the dependencies among various components
  - Use of redundant mechanisms where plausible
- Break the "routing" up into a number of subsystems, with the interfaces between them visible, *and available*, to the users.

Nimrod Architecture Overview

Nimrod Fundamental Architecture

The Nimrod routing architecture has the following key characteristics:

- Map Distribution type
- Explicitly routed

This combination has a number of advantages:

- Allows maximal overall flexibility (i.e. system lifetime)
- Is securable against explicit attack
- Allows maximal robustness
- Interacts well with flexible abstraction mechanisms
- Allows policy routing (allows complete source control over packet forwarding path).
- Allows maximally flexible policy routing
- Interacts well with resource allocation systems

Nimrod Fundamental Architecture, Continued

It also includes the following key concepts:

- The fundamental mechanism for forwarding user data packets is the *flow*
- Uses hierarchical abstraction to allow and manage scaling

Note: The path-selection algorithm is ***not*** part of the protocol (although a sample algorithm can be given in an appendix).

Additional Nimrod Architectural Points

The Nimrod routing architecture is also characterized by the following concepts:

- Provision for efficient handling of datagram service mode packets, using a mesh of Datagram Mode Flows (DMF's) which are maintained at all times
- Flexible abstraction techniques:
  - Compression + Thinning $\Longrightarrow$ Abstraction
  - Thinning is needed for practicality, but causes non-optimal routes
  - Nimrod allows users to pick their own balance between these two costs
  - Abstraction algorithms (outgoing and incoming) are not part of the spec
- Complete routing system, down to the lowest levels, with no simplistic IGP/EGP split

Reflections on the Architecture

- Note what is ***not*** part of the architecture:
  - Path-selection algorithm
  - Outgoing abstraction algorithm
  - Incoming abstraction algorithm ("Local Abstraction Control")
- Key advantages of this approach:
  - Limits amount of work to be done
  - Allows future improvement
  - Allows easy deployment of new algorithms
  - Reduces scope for errors and bad design by minimizing global mechanisms
- What ***is*** part of the architecture:
  - Method of representing topology
  - Method of distributing topology information
  - Method of setting up flows, and handling datagram mode packets