   Architecture for Integrated Directory Services – Result from TISDAG

Status of this Memo

Copyright Notice

Abstract

   A single, unified, global whitepages directory service remains
   elusive.  Nonetheless, there is increasing call for participation of
   widely-dispersed directory servers (i.e., across multiple
   organizations) in large-scale directory services.  These services
   range from national whitepages services, to multi-national indexes of
   WWW resources, and beyond.  Drawing from experiences with the TISDAG
   (Technical Infrastructure for Swedish Directory Access Gateways)
   ([TISDAG]) project, this document outlines an approach to providing
   the necessary infrastructure for integrating such widely-scattered
   servers into a single service, rather than attempting to mandate a
   single protocol and schema set for all participating servers to use.

1. Introduction

   The TISDAG project addressed the issue of providing centralized
   access to distributed information for whitepages information on a
   national scale.  The specification of the eventual system is
   presented in [TISDAG], and [DAGEXP] outlines some of the practical
   experience already gained in implementing a system of this scale and
   nature.  [DAG-Mesh] considers the issues and possibilities of
   networking multiple DAG services.  Following on from those, this
   document attempts to describe some of the architectural underpinnings
   of the system, and propose directions in which the approach can be
   generalized, within the bounds of applicability.

   The proposed architecture inserts a coordinated set of modules
   between the client access software and participating servers.  While
   the client software interacts with the service at a single entry
   point, the remaining modules are called upon (behind the scenes) to
   provide the necessary application support.  This may come in the form
   of modules that provide query proxying, schema translation, lookups,
   referrals, security infrastructure, etc.

   Part of this architecture is an "internal protocol" -- called the
   "DAG/IP" in the TISDAG project.  This document also outlines the
   perceived requirements for this protocol in the extended DAG.

2.0 Some terminology

   Terms used in this document are compliant with those set out in
   [ALVE]. For the purposes of this document, important distinctions and
   relationships are defined between applications, services, servers and
   systems.  These are defined as follows:

   Application:  this is meant in the general sense, as a solution to a
      particular (set of) user need(s).  That is, the definition is not
      tied to a particular piece of software (as in "application
      program").

      The definition of an application includes the type(s) of
      information to be exchanged, expected behavior, etc.  Thus, a
      whitepages (search) application may expect to receive a name as
      input to a query engine, and will return all information associated
      with the name.  By contrast, a specific security application might
      use the same input name to verify access controls.

   Service:  an operational system providing (controlled) access to
      fulfill a particular application's needs.

      One service may be changed by configuring location, access
      controls, etc.  Changing application means changing the service.

   Server:  a single component offering access through a dedicated
      protocol, without regard to a specific service (or services) it may
      be supporting in a given configuration. Typically programmed for a
      particular application.

   System:  a set of components with established interconnections.

      Thus, a service can be split between several servers.  A collection
      of services (independently, or interrelated through specified
      agreements) act as an implementation of an application.  A system
      is composed of one or more servers and services.

A "system architecture" identifies specific software components, their behavior, communication channels and messages needed to fulfill a particular service's needs.  The TISDAG specification [TISDAG] includes just such a description, defining a software system that will meet the needs of a national whitepages directory service.  Here, we outline some of the general principles which lead to that specific system architecture and discuss ways in which the principles can be applied in other contexts.

Looking at this bigger picture, we present a "service architecture", or a framework for assembling components into systems that meet the needs of a wider variety of services.  This is not a question of developing one or more new protocols for services, but rather to examine a useful framework of interoperating components.  The goal is to reduce the overall number of (specialized) protocols that are developed requiring incorporation of some very general concepts that are common to all protocols.

3.0  TISDAG -- a first implementation, and some generalizations

The Swedish TISDAG project (described in detail in [TISDAG], with some experiences reported in [DAGEXP]) was designed to fulfill the requirements of a particular national directory service.   The experience of developing component-based system for providing a directory service through a uniform interface (client access point) provided valuable insight into the possibilities of extending the system architecture so that services with different base requirements can benefit from many of the same advantages.

3.1 Deconstructing the TISDAG architecture

In retrospect, we can describe the TISDAG system architecture in terms of 3 key requirements and 4 basic design principles:

   R1. The service had to function with (several) existing client and
       server software for the white pages application.

   R2. It had to be possible to extend the service to accommodate new
       client and server protocols if and when they became relevant.

   R3. The service had to be easily reconfigurable -- to accommodate
       more machines (load-sharing), etc.

   D1. As a design principle, it was important to consider the
       possibility that queries and information templates (schema)
       other than the originally-defined set might eventually be
       supported.

   D2. As the architecture was already modular and geared towards
       extensibility, it seemed important to keep in mind that the
       same (or a similar) system could be applied to other (non-
       white pages) applications.

   D3. There is an "inside" and an "outside" to the service --
       distinguishing between components that are accessible to the
       world at large and those that are open only to other
       components of the system.

   D4. Internally, there is a single protocol framework for all
       communications -- this facilitates service support functions
       (e.g., security of transmission), ensures distributability,
       and provides the base mechanism for allowing/ascertaining
       interoperability of components.

   The resulting system architecture featured modular component (types)
   to fulfill a small number of functional roles, interconnected by a
   generic query-response language.  The functional roles were defined
   as:

      CAPs -- "client access points" -- responsible for accepting and
      responding to incoming requests through programmed and configured
      behavior -- to translate the incoming query into some set of DAG-
      internal actions (queries) and dealing with the responses,
      filtering and recombining them in such a way as to fulfill the
      client request within the scope of the service.  In the TISDAG
      system, all CAPs are responsible for handling whitepages queries,
      but the CAPs are distinguished by the application protocol in
      which they will receive queries (e.g., LDAPv2, LDAPv3, HTTP, etc).
      To the client software, the TISDAG system appears as a server of
      that particular protocol.  In the more general case, CAPs may be
      configured to handle different aspects of a service (e.g.,
      authenticated vs.  non-authenticated access).  While the TISDAG
      CAPs all had a simple control structure, the more general case
      would also see CAPs drawing on different subsets of DAG (internal)
      servers in order to handle different query types.  (See the
      "Operator Service" example, in section 5.2 below).

      SAPs -- "service access points" -- responsible for proxying DAG-
      internal queries to specified services.  These are resources drawn
      upon by other components within the system.  Through programmed
      and configured behavior, they translate queries in the internal
      protocol into actions against (typically external) servers, taking
      care of any necessary overhead or differences in interaction
      style, and converting the responses back into the internal
      protocol.  In the TISDAG system, all SAPs are responsible for
      handling whitepages queries, but they are distinguished by the

      application protocol in which they will access remote services.
      Further distinctions could be made based on the (remote service's)
      schema mappings they handle, and other service differentiators.

      Internal Servers respond to queries in the internal protocol and
      provide specific types of information.  In the TISDAG system,
      there is one internal server which provides referral information
      in response to queries.

   Note that all these components are defined by the functional roles
   they play in the system, not the particular protocols they handle, or
   even the aspect of the service they are meant to support.  That is, a
   client access point is responsible for handling client traffic,
   whether its for searching, establishing security credentials, or some
   other task.

3.2 Some generalizations

   The Requirements and Design principles outlined above are not
   particular to a national whitepages service.  They are equally
   applicable in any application based on a query-response model, in
   services where multiple protocols need to be supported, and/or when
   the service requires specialized behavior "behind the scenes".  In
   the TISDAG project, this last was inherent in the way the service
   first looks for referrals, then makes queries as appropriate.  For
   protocols that don't handle the referral concept natively, the TISDAG
   system proxies the queries.

   Because of its particular application to query-response situations,
   the term "Directory Access Gateway", or "DAG" still fits as a label
   for this type of system architecture.

   Internet applications are evolving, and require more sophisticated
   features (e.g., security mechanisms, accounting mechanisms,
   integration of historical session data).  Continuing to develop a
   dedicated protocol per application type results in encumbered and
   unwieldy protocols, as each must implement coverage of all of these
   common aspects.  But creating a single multi-application protocol
   seems unlikely at best.  The implicit proposal here is that, rather
   than overloading protocols to support multiple aspects of a service,
   those aspects can be managed by breaking the service into multiple
   supporting components to carry out the specialized tasks of
   authentication, etc.

3.3 A Word on DAG/IP

   In the TISDAG project, the choice was made to use a single "internal
   protocol" (DAG/IP).  The particular protocol used is not relevant to
   the architecture, but the principle is important.  By selecting a
   single query-response transaction protocol, the needs of the
   particular application could be mapped onto it in terms of queries
   and data particular to the application.  This makes the internal
   communications more flexible for configuration to other environments
   (services, applications).

   It is common today to select an existing, widely deployed protocol
   for transferring commands and data between client and server -- e.g.,
   HTTP.  However, apart from any issues of the appropriateness (or
   inappropriateness) of extending HTTP to this use, the work would have
   remained to define all the transaction types and data types over that
   protocol -- the specification of the interaction semantics and
   syntax.

3.4 Perceived benefits

   Apart from the potential to divide and conquer service aspects, as
   described above, this approach has many perceived benefits:

      - For multi-protocol environments, it requires on the order of
        N+M inter-protocol mappings, not NxM.
      - distribution of development
      - distribution of operation
      - eventual possibilities of hooking together different
        systems (of different backgrounds)
      - separation of
              - architectural principles
              - implementation to a specific application
              - configuration for a given service

   It is not the goal to say that a standardized system architecture can
   be made so that single components can be built for all possible
   applications.  However, this approach in general permits the
   decoupling of access protocols from specific applications, and
   facilitates the integration of necessary infrastructure independently
   of access protocol (e.g., referrals, security, lookup services,
   distribution etc).

4.0 Proposed service architecture

    Pictorially, the DAG architecture is as follows:

```
        +-------------------------------------------+
   "a" |                  |                         |
<----->|   CAP a          |       +--------+        |
       |                  |       | SAP A  |        |
       |                  |       |        |        |
       |--------+         |     +-+------+---+       |
       |                  |     | (Internal) |      |
       |         "DAG/IP" |     |  Server i  |      |
       |                  |     +----------+        |
       |                  |                         |
       |                  |                         |
       |                  |                         |
       |                  |     +--------+   |  "B" |
       |                  |     | SAP B  <--------------->
       |                  |     |        |   |      |
       |                  |     +--------+   |      |
       |                  |                         |
        +-------------------------------------------+
```
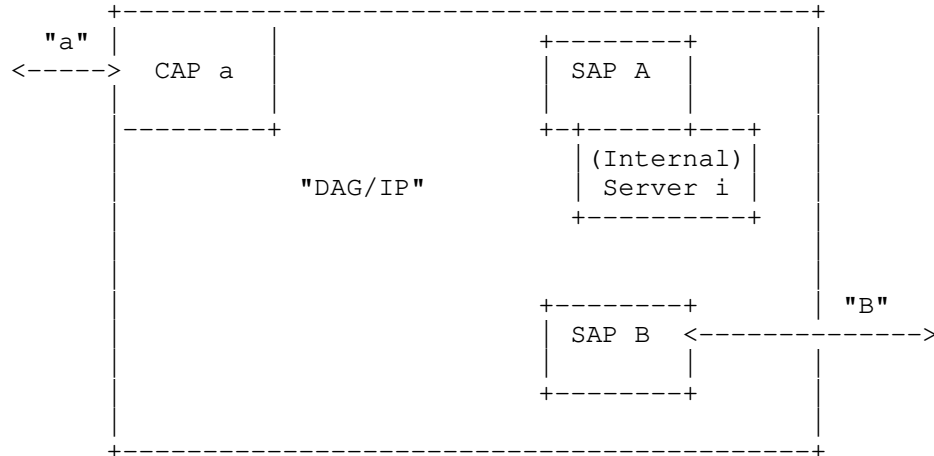
    Note that the bounding box is conceptual -- all components may or may
    not reside on one server, or a set of servers governed by the
    provider of the service.

    As we saw in the TISDAG project, the provider of this DAG-based
    service may be only loosely affiliated with the remote services that
    are used (Whitepages Directory Service Providers (WDSPs) in this
    case).

4.1 Using the architecture

    Building a service on this architecture requires:

    Service implementation:
       1. definition of the overall application to be supported by the
          system -- whitepages, web resource indexing, medical
          information
       2. requirements
       3. expected behavior

    System architecture:
       1. nature of deployment -- distributed, security requirements,
          etc.
       2. identification of necessary CAPs -- in terms of access
          protocols to be supported, different service levels to be
          provided (e.g., secure and unsecure connections)

     3. identification of necessary services -- e.g., proxying to
       remote information search services, lookup services, "AAA[A]"
       (Authentication, Authorization, Accounting, [and Access])
       servers, etc
     4. definition of the transaction process for the service:  insofar
       as the CAPs represent the service to client software, CAP
       modules manage the necessary transactions with other service
       modules

  Data architecture:

     1. selection of schemas to be used (in each protocol)
     2. definition of schema and protocol mappings -- into and out of
       some DAG/IP representation

## 5.0  Illustrations

## 5.1 Existing TISDAG Project

  Consider the TISDAG project in the light of the above definitions.

  Service implementation:
     1. A national-scale subset of Whitepages lookups, with specific
       query types supported: only certain schema attributes were
       permitted in queries, and the expected behavior was limited in
       scope.
     2. Requirements: the service had to support multiple query
       protocols (from clients and for servers), and be capable of
       searching the entire space of data without centralizing the
       storage of records.
     3. Given a query of accepted type, provide referrals to whitepages
       servers that might have information to fulfill the query; if
       necessary, proxy the referrals (chain) to retrieve the
       information for the client.

  System architecture:
     1. distributable components
     2. publicly accessible CAPs in HTTP, SMTP, Whois++, LDAPv2, and
       LDAPv3
     3. referral proxies to Whois++, LDAPv2 and LDAPv3 WDSPs, as well
       as a referral query service
     4. the basic transaction process, uniform across all CAPs, is:
          - query the RI for relevant referrals
          - where necessary, chain referrals through SAPs of
            appropriate protocol return, in the native protocol, all
            remaining referrals and data

    Data architecture:  see the spec.

    In the TISDAG project, the above diagram could be mapped as follows:

        CAP a           LDAPv2 CAP
        SAP A           the Referral Index (RI) interface
        Server i        the Referral Index (RI)
        SAP B           LDAPv3 SAP

    Note that, in the TISDAG project specification, the designation SAP
    referred exclusively to proxy components designed to deal with
    external servers.  The Referral Index was considered an entity in its
    own right.  However, generalizing the concepts of the TISDAG
    experience lead to the proposal of regarding all DAG/IP-supporting
    service components as SAPs, each designed to carry out a particular
    type of service functionality, and whether the server is managed
    internally to the DAG system or not is immaterial.

5.2 Operator service

    Consider the case of "number portability" -- wherein it is necessary
    to determine the current service provider of a specific phone number.
    The basic assumption is that phone numbers are assigned to be
    globally unique, but are not in any way tied to a specific service
    provider.  Therefore, it is necessary to determine the current
    service provider for the given number before being able to retrieve
    current information.  For the sake of our illustration, let us assume
    that the management of numbers is two-tiered -- suppose the system
    stores (internally) the mapping between these random digit strings
    and the country in which each was originally activated, but relies on
    external (country-specific) services to manage the updated
    information about which service provider currently manages a given
    number.  Then, the service data need only be updated when new numbers
    are assigned, or national services change their access points.

    We can look at a grossly-simplified version of the problem as an
    illustration of some of the concepts proposed in this service
    architecture.  We couple it with the "name search" facet of the
    TISDAG example, to underscore that a single service ("operator") may
    in fact be supported by several disjunct underlying activities.

    Service implementation:
      1. Retrieving service information for a particular (unstructured)
         phone number digit sequence, or searching for numbers
         associated with a particular name (or fragment thereof).
      2. Requirements:  support IP-telephony through HTTP-based
         requests, wireless device requests through WAP [WAP].

       3. Expected behavior:  given a name (fragment), return a list of
          names and numbers to match the fragment; given a phone number,
          return appropriately-structured information re. the current
          service mapping for that number.

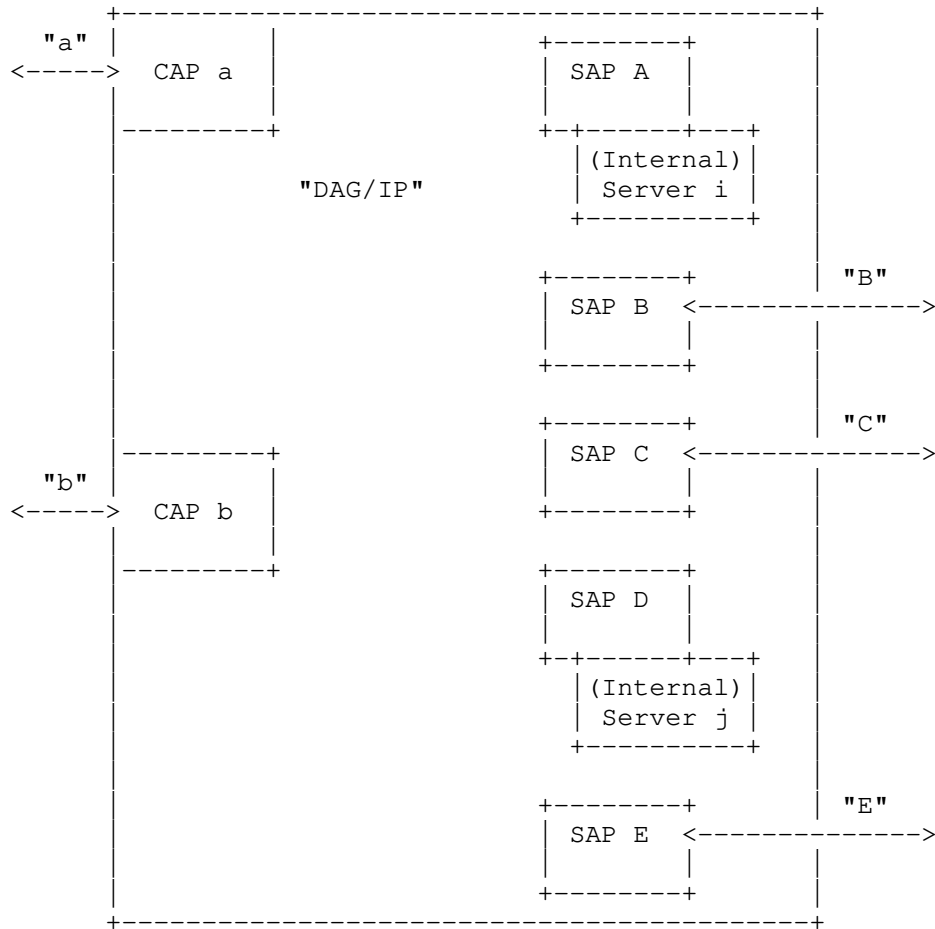    System architecture:
       1. Publicly accessible through CAPs; components widely
          distributed.
       2. Need one CAP for HTTP, one for WAP.
       3. Support services include:  an internal service for lookup of
          number strings (to identify nation of origin of the number), a
          proxy to access national services for registration of numbers
          and service providers, and a proxy for remote service provider
          for retrieval of detailed information regarding numbers.  For
          the name searching, we also need a referral index over the
          names, and a proxy to whatever remote servers are managing the
          whitepages directories.
       4. Now, 2 different types of transaction are possible:  search for
          name, or look-up a number.  In the name search case, the CAP
          receives a name or name fragment, looks it up in the internal
          referral index, and finds associated numbers through external
          whitepages services (WDSPs).  To look-up a number, the CAP
          first uses the internal look-up service to determine the
          country of origin of the number, and then uses a SAP to access
          that nation's number-service provider directory, and finally
          uses a different SAP to access the current service provider to
          determine the information required to make the call.

    Data architecture:
       [Out of scope for the purposes of this illustration]

       Note that some elements of the system architecture are
       deliberately vague.  Per the requirements, no structure is
       expected in the number string, and therefore the lookup server
       must maintain an index of number-to-country mappings and relies
       on an external number-to-service mapping service (in each
       country).  However, were there any structure to the numbers, the
       lookup server could make use of that structure in the indexing,
       or in distribution of the index itself.  This would have no
       effect on the CAPs, which have no inherent reliance on how the
       lookup server performs its task.

Pictorially, the example can be rendered as follows:

```
          +-------------------------------------+
   "a" |                        +--------+      |
<-----> |  CAP  a |             | SAP A  |      |
        |         |             |        |      |
        |---------+             +-+------+---+   |
        |                         |(Internal)|   |
        |           "DAG/IP"      | Server i |   |
        |                         +---------+    |
        |                                        |
        |                       +--------+   "B" |
        |                       | SAP B  <-------------->
        |                       |        |      |
        |                       +--------+      |
        |                                        |
        |                       +--------+   "C" |
        |---------+             | SAP C  <-------------->
   "b" |         |              |        |      |
<-----> |  CAP  b |             +--------+      |
        |         |                             |
        |---------+             +--------+      |
        |                       | SAP D  |      |
        |                       |        |      |
        |                       +-+------+---+   |
        |                         |(Internal)|   |
        |                         | Server j |   |
        |                         +---------+    |
        |                                        |
        |                       +--------+   "E" |
        |                       | SAP E  <-------------->
        |                       |        |      |
        |                       +--------+      |
        +-------------------------------------+
```

where

```
     CAP a          HTTP CAP
     CAP b          WAP CAP
     SAP A          the number-nation lookup interface
     Server i       number-nation lookup server (what country)
     SAP B          nation-service lookup SAP (what service provider)
     SAP C          service-number information lookup SAP (current
                    service details)
     SAP D          referral index interface
     Server j       referral index service
     SAP E          proxy for chaining queries to remote WDSPs
```

5.3 Medical application

   The service architecture is useful for applications outside the scope
   of "telecoms".  In another hypothetical illustration, consider the
   case of medical information -- records about patients that may be
   created and stored at a variety of institutions which they visit.  It
   is not unusual to need to access all information concerning a
   patient, whether or not the person can recollect (or communicate)
   conditions that were treated, procedures that were performed, or
   medical institutions visited.  The data may include everything from
   prescriptions, to X-rays and other images, to incident reports and
   other elements of medical history, etc.  Typically, the information
   is stored where it is collected (or by an agency authorized by that
   institution) -- not in a central repository.  Any service that looks
   to provide complete answers to queries must deal with these
   realities, and clearly must function with a strong security model.

   Service implementation:
      1. Retrieving all medical information for a particular person.
      2. Requirements:  must retrieve, or at least locate, all
         available information, regardless of its storage location;
         cannot require central repository of information; must
         implement authorization and access controls.  Must
         support a proprietary protocol for secure connections
         within hospitals, wireless access for personnel in
         emergency vehicles (not considered secure access).
      3. Expected behavior:  given a patient's national ID, and
         authorized access by medical personnel in secure locations,
         determine what kinds of records are available, and where;
         given a request for a specific type of record, retrieve
         the record.  Given a patient's national ID, and authorized
         access from a wireless device, provide information re.
         any known medical flags (e.g., medicine allergies,
         conditions, etc).

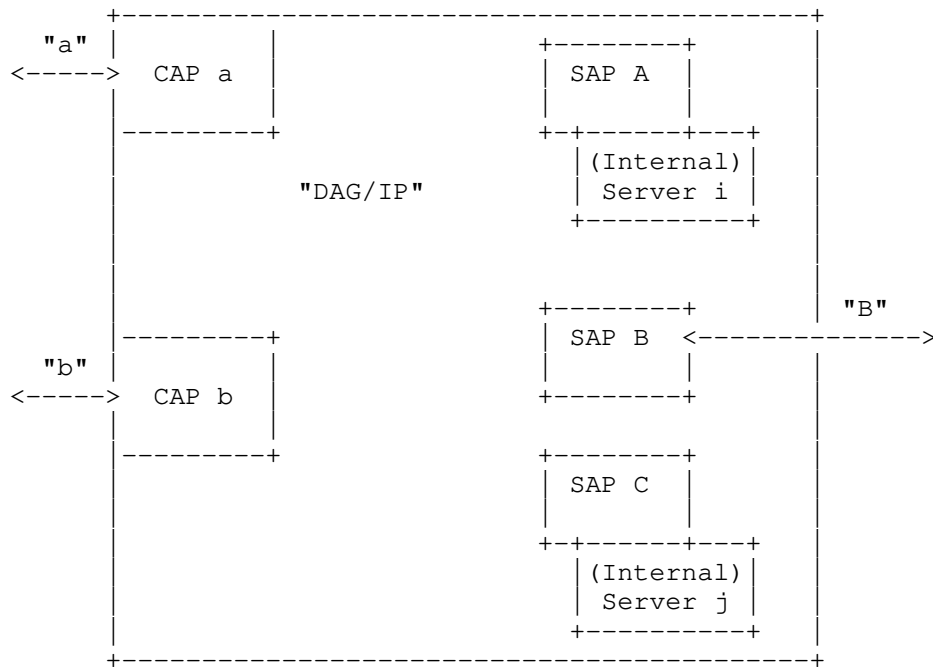   System architecture:
      1. Only 2 CAP types are needed, but instances of these will
         be established at major medical institutions.
      2. Need one CAP to support the proprietary protocol, one
         to support wireless access.
      3. Support services include:  an internal server to support
         security authentication and access control determination;
         an internal server to act as referral index for finding
         information pertinent to a particular patient, and one
         or more proxies for accessing remote data storage servers.
      4. The basic transaction requires that the first step be
         to authenticate the end-user and determine access privileges.
         In the case of wireless access, this last will not involve

          a specific lookup, but rather will be set to allow the
          user to see the list of publicized medical conditions.
          Depending on the query type, the next step will be to
          contact the referral index to determine what records
          exist, and then track down information at the remote sources.

   Data architecture:
          [Out of scope for the purposes of this illustration]

   Pictorially, the example can be rendered as follows:

```
         +------------------------------------------+
   "a"   |                         +--------+       |
<----->  |   CAP a |               | SAP A  |       |
         |         |               |        |       |
         |         |               |        |       |
         |---------+               +-+------+---+   |
         |                           |(Internal)|   |
         |            "DAG/IP"       | Server i |   |
         |                           +----------+   |
         |                                          |
         |                                          |
         |                         +--------+  "B"  |
         |---------+               | SAP B  <-------------->
   "b"   |         |               |        |       |
<----->  |   CAP b |               +--------+       |
         |         |                                |
         |---------+               +--------+       |
         |                         | SAP C  |       |
         |                         |        |       |
         |                         +-+------+---+   |
         |                           |(Internal)|   |
         |                           | Server j |   |
         |                           +----------+   |
         +------------------------------------------+
```

   where

        CAP a          CAP for proprietary protocol, secure clients
        CAP b          WAP CAP, for roaming access
        SAP A          authentication and ACL lookup interface
        Server i       authentication and ACL lookup server
        SAP B          remote service SAP -- probably LDAPv3
        SAP C          Referral Index interface
        Server j       Referral Index

6. Requirements for the future DAG/IP

   The role of the DAG/IP is less as a query protocol, and more as a
   framework or structure for carrying basic query-response transactions
   of different (configurable) types.

   Whatever the syntax or grammar, the basic requirements for the DAG/IP
   include that it be:

      - lightweight; CAPs, SAPs should be able to be quite small
      - flexible enough to carry queries of different paradigms, results
        of different types
      - able to support authentication, authorization, accounting and
        audit mechanisms -- not necessarily native to the protocol
      - able to support encryption and end-to-end security within the
        DAG system
      - sophisticated enough to allow negotiation of  capabilities --
        querying & identifying application type supported (e.g.,
        whitepages vs. service location vs. URN resolution), query types
        supported, results types supported

      This also means:

   Better support for query-passing/other query semantics (need to
   balance that against the fact that you don't want DAG-CAPs/SAPs to
   have to know a multiplicity of semantic possibilities.

   Security infrastructure -- ability to establish security credentials,
   maintain a secure transaction, and propagate the security information
   forward in the transaction (don't want to reinvent the wheel, just
   want to be able to use it!).

   Ability to do lookups, instead of searches -- might mean connecting
   to different services than the RI and/or presenting things in a
   slightly different light -- e.g., lookup <blat> in the <foo> space,
   as opposed to search for all things concerning <blat>.

   Ability to access other services -- e.g., Norwegian Directory of
   Directories [NDD] -- beyond just for specific characteristics of the
   service (e.g., security).

   In short, the model that seems to stand out from these requirements
   one of a protocol framework that looks after establishing secure and
   authenticated (authorized, accountable, auditable...) connections,
   with transaction negotiation facilities.  Within that framework, it
   must be possible to identify transaction types, provide suitable
   input information (negotiation?) for those transactions, and accept
   transaction result objects back.

7. Revisiting TISDAG -- for the future

   In the light of the above proposals, we can revisit the way the
   TISDAG CAPs would be defined.

   The whitepages-application service known as TISDAG could have SAPs
   that supported 2 types of query, and 2 types of result sets:

           query types:
                   . token-based
                   . phrase-based

           result types:
                   . result data
                   . referrals

   The Whois++ CAP would be configured to contact LDAPv2 and LDAPv3 SAPs
   because they are identified as providing that kind of service (i.e.,
   if referral protocol == LDAPv2 connect to a particular service).  The
   query paradigm will be phrase-oriented -- NOT because the Whois++ CAP
   understands LDAP, but because that is one of the defined query types.

8. Applicability Limitations

   As it stands, this type of service architecture is limited to query-
   response type transactions.  This does account for a broad range of
   applications and services, although it would be interesting to
   consider broadening the concept to make it applicable to tunneling
   other protocols (e.g., to connect a call through a SAP, in the number
   portability example above).

9. Security Considerations

   This document takes a high-level perspective on service architecture,
   and as such it neither introduces nor addresses security concerns at
   an implementation level.

   A distributed service built following this approach must address
   issues of authentication of users, authorization for access to
   material/components of the system, and encryption of links between
   them, as befits the nature of the information and service provided.

10. Acknowledgements

   In discussing this perspective on the evolution of DAG/IP, it seemed
   to us that the requirements for DAG/IP are falling into line with the
   proposed text-based directory access protocol that has variously been
   discussed.  Whether it survives in a recognizable form or not :-)
   some of the above has been drawn from discussions of that protocol
   with Michael Mealling and Patrik Faltstrom.

   The work described in this document was carried out as part of an on-
   going project of Ericsson.  For further information regarding that
   project, contact:

      Bjorn Larsson
      bjorn.x.larsson@era.ericsson.se

11. Authors' Addresses

   Leslie L. Daigle
   Thinking Cat Enterprises

   EMail:  leslie@thinkingcat.com


   Thommy Eklof
   Hotsip AB

   EMail: thommy.eklof@hotsip.com

12. References

   Request For Comments (RFC) and Internet Draft documents are available
   from numerous mirror sites.

   [ALVE]     Alvestrand, H., "Definitions for Talking about
              Directories", Work in Progress.

   [TISDAG]   Daigle, L. and R. Hedberg "Technical Infrastructure for
              Swedish Directory Access Gateways (TISDAG)", RFC 2967,
              October 2000.

   [DAGEXP]   Eklof, T. and L. Daigle, "Wide Area Directory Deployment
              Experiences", RFC 2969, September 2000.

   [DAG-Mesh] Daigle, L. and T. Eklof, "Networking Multiple DAG servers:
              Meshes", RFC 2968, September 2000.

RFC 2970        Architecture for IDS – Result from TISDAG     October 2000

      [NDD]        Hedberg, R. and H. Alvestrand, "Technical Specification,
                   The Norwegian Directory of Directories (NDD)", Work in
                   Progress.

      [WAP]        The Wireless Application Protocol, http://www.wapforum.org

13.  Full Copyright Statement

Acknowledgement