Naming and Information Structures, Rendezvous Security, and Data-Centric Routing
T-110.6120 17.2.2010

Kari Visala
Helsinki Institute for Information Technology HIIT /
Helsinki University of Technology TKK
Different Types of Pub/Sub

- Content-based
  - Subscribers can subscribe to all publications based on their content. Typically, each publication has a set of (attribute name, value) pairs and subscriptions define filters on possible values of different attributes.

- Topic-based
  - Messages are published to named topics and subscribers receive all publications targeted to the topics of their interest.

- Data-centric
  - Individual data objects are named and subscribed individually.
Dual Paradigms

Data-centric

Identifier

Value

Message-oriented

Identifier

Continuation

Concepts

Implementation

receive data from <id,address>

Server

Data

send <id, data> to address

Server

Session state
Adding the Missing Half of the Internet

- A data-centric network layer would have many advantages:
  - Lower latency because of caching
  - Native asynchronous multicast → efficiency, no flash crowd bottleneck
  - Less layers
  - No unwanted traffic
- P2P overlays have efficiency, incentive, and security problems
- Patching Internet with CDNs is also a bit unsatisfactory
  - interworking of different systems, adaptation to demand generated by subscribers, limited to web semantics, optimality of network resource allocation, unwanted traffic still possible
- → Lots of research: CCN, DONA, ROFL, TRIAD, ..
- A clean-slate pub/sub network architecture is being developed in the PSIRP EU project
  - Bloom filter based forwarding [Jokela et al. 2009]
  - Hierarchical DHT-based rendezvous architecture [PSIRP TR09-003]
  - In this paper we try to bridge these two layers with an intermediate routing layer
General Design Principles

- End-to-end, Neutrality
  - Network layer should be seen as a minimal enabler that is mostly concerned about network resource allocation.
  - Push as much functionality to the higher layers as possible.
- "Low-level"
  - Network semantics should not be made transparent to upper layers if they really are different from local communication
    - Partial failure, security etc.
- Universality
  - Data-centric instead of content-based: *optimizations* should not be arbitrarily added on the network layer if they interfere with the evolution of the system
    - Van Jacobson: "Google goes from content to names"
Basic Concepts
Identifiers and Namespaces

- Identifiers can name anything!
  - Scopes, publications, continuations, mutable objects, types ..
- DONA-like <P,L> structure for network level identifiers
  - P is the public key of namespace owner
    - Trust anchor for checking the integrity of the publication value.
    - Authorizes publishers to use parts of the namespace with a signed certificate that could contain the possible names as a regexp.
    - No tussle for global human-readable names on this level.
  - L is a variable length label
    - Label can contain semantic information that network should not interpret \(\rightarrow\) dynamic data.
    - Binary data that can encode character strings.
    - Only fixed length hash can be used in payload packets.
- Because security implementation is stored in the identifier, they can only have short lifetime.
- Application level names
  - Name application level constructs, typed.
  - Use DNS, for example, to map namespace name to namespace owner P \(\rightarrow\) long lifetime and can be fully human-readable. Multiple mechanisms are possible.
  - Many schemes to map application constructs to network publications (alg. ids).
Scopes

- Scopes adhere to the "low-level" philosophy.
  - For example, without scopes, how could we prevent fake data sources?
  - They make the network more complicated to use – a kind of "abstract location".
  - On the other hand, we should not hide real complexity of policies etc. from the applications.
- Scopes represent *distribution strategies*.
  - Orthogonal to the structure of the information
  - Controls, among others, access rights of data sources and subscribers, location, reachability, availability, replication, persistence, and upstream resources and policies
  - Many types of scopes: P2P?
- Scopes are abstract
  - Realized by authorized data sources and home rendezvous networks of the scope.
Publications (on the application layer)

- Publication is a *immutable association with the identifier and the value of the publication* after it has been created/published.
  - Resembles *static single assignment* (SSA) that is used in some compilers.
  - Content can always be cached and used locally if found without synchronization problems.
- Publisher and data source are two separate entities that are not necessarily related.
- It is assumed that the client knows which identifier she wants and the *secure association* can be done between the identifier and the value of the publication.
- Publications can be typed (on the application layer).
- Publications can be dynamic (content is created on-demand)
  - Explicit *futures*: `Future<T>` where `T` is the type of the publication
  - Can be generalized to nullary functions such as label "Weather(Sofia, 18.1.2010)"
    - Similar to *thunks* used in the implementation of lazy functional programming languages. Could the network be seen as a single, large evaluation environment for functional programs?
- Application level publications are split into multiple network level publications (with their own algorithmic ids computed at the client)
An Example

- Application level publication:
  - Aid
    - ("fmi.fi", DNS, "weather(Sofia, 18.1.2010)")
  - Application level scope name
    - ("google.com", DNS, "Sonera secure scope")
  - Type id could point to a definition:
    - WeatherType = Sunny | Raining

- Subscriber translates Aid and scope into (Sid, Rid) pair to retrieve the current weather:
  - Rid
    - (<current public key of Finnish meteorological institute namespace>, "weather(Sofia, 18.1.2010)")
  - Sid
    - (<current public key of Google namespace>, "Sonera scope")

- Value returned by the network:
  - Sunny

- Both signature chain and type are checked before using.
Publication Granularity

- For efficiency, incremental modifications to large publications would be preferable
  - → application level publications are not directly network level
- We may support even higher granularity with lightweight segmentation on the network layer → does not directly help with the partial modifications
  - More efficient as not every segment needs to have a name of its own
  - On the other hand, such optimizations always increase the complexity in a biased way
Emulation of Mutable State / 1 \(\rightarrow\) N Channels

- Immutable publications (instead of naming "memory addresses") free us from thinking about synchronization on lower layers and lead to functional programming style.

- Mutable state could be implemented using a time based (algorithmic) id that returns the current version number of the publication.
  - "avatar coordinate at 14:34:33 on 18.1.2010" \(\rightarrow\) "version 2432"
  - After that, the subscriber can subscribe to subsequent (algorithmic) labels "avatar coordinate version 2432", ".. version 2433" etc.

- 1 \(\rightarrow\) N channels can be emulated in a similar way.

- Drawback: longer initial latency as the version number of the current publication needs to be first fetched

- Similarly, different runtime instances of the same application need to use different versions for all publications that value can change.
Emulation of Message Passing

- Message passing can be emulated using dynamic labels:
  - Subscriber $X$ subscribes to dummy label "$X$ offers data $Y$"
  - Data source may acknowledge by returning "ok" as the value of the subscription.
  - The original data source subscribes $Y$.
- Drawback: initial latency of roundtrip time.
Emulation of N $\rightarrow$ M Events

- Aggregator in the middle: Combine $N \rightarrow 1$ with $1 \rightarrow N$
  - Clear ownership $\rightarrow$ security
  - Clear semantics for filtering excess events
  - Additional latency
Phases of Communication

1. A set of potential Rlds for a scope is advertised in the HRN of the scope.

2. Scope X is advertised in the rendezvous interconnect

3. Subscriber rendezvous with the SId:Rld

4. Subscription message accumulates an Fld to the reverse direction

5. Publication data is delivered to the subscriber using Fld
Rendezvous Network

- Each rendezvous network hosts a set of scopes.
- Data sources of a scope can advertise the set of (potential) RIds that they can serve.
  - The home rendezvous network can itself be found globally using the same rendezvous mechanism.
  - The protocol between the data source and the home rendezvous network of the scope can be proprietary and support different representations for a set of publications.
    - For example, a rendezvous network could function as the tracker for a BitTorrent like file distribution.
- A bilateral contract is assumed between the scope and the rendezvous network.
  - The rendezvous network can be compensated for the use of its resources → persistent advertisements.
- DONA is a possible implementation for a large scale rendezvous network spanning multiple domains.
DONA as a Rendezvous Network

• In DONA system data is advertised in every domain in the *upgraph* of the publisher.

• Subscriber can simply find a policy-compliant path by searching for the advertisement by recursively querying its providers in a bottom-up fashion.

• (image by Jarno Rajahalme)
2-Tier Architecture

image by Jarno Rajahalme
Rendezvous Interconnect Architecture

- DONA is not scalable for global-level rendezvous as it stores a copy of all advertisements in all tier-1 domains.
- ROFL uses a DHT to achieve scalability, but the peer-to-peer nature of the system creates incentive problems.
- We also want to support specialized rendezvous networks (access control, different mechanisms for registering publications etc.).
- Therefore, we should use a 2-tier system where a hierarchical DHT based rendezvous interconnect network joins multiple rendezvous networks together for global reachability.
  - Typically only scopes are advertised in the interconnect.
  - Hierarchical structure guarantees *locality* for the communication.
- Local rendezvous networks and rendezvous interconnect nodes can cache results for individual public (SId, RId) pairs and subscribe to the changes by forming a multicast tree using the DHT routing alg.
- Rendezvous is not required for each individual publication from the same scope but the subscriber can opportunistically try to subscribe them from a known data source.
- Rendezvous interconnect has only *soft state*, that can be recovered by the rendezvous networks.
Canon (Hierarchical Chord DHT)

- A hierarchical structure of organizations is assumed.
  - Each subhierarchy forms a Chord ring of its own.
  - When two rings are merged, an additional finger from node $m$ to $m'$ in the other ring is created if and only if $m'$ is the closest node that is at least distance $2^k$ away form some $0 \leq k < N$ and $m'$ is closer to $m$ than any other node in $m'$s ring.
High-Level Security Goals for the Rendezvous Interconnect

• Main Goals
  • Availability
  • Fairness
  • (Data) Integrity

• Additional goals
  • Confidentiality
  • Anonymity
  • Accountability
Attack Types

- Interfering with the DHT routing, storage, and forwarding algorithm
- Attacks based on using the system
  - DDoS
  - Consuming more or providing less than fair share of resources
- Content poisoning and enumeration
- Discovering node identities and connecting them to lookup/store operations
- General attacking methods
  - Sybil attack
  - Eclipse attack
- Attacks against the underlaying network
  - Compromised link
  - DDoS
Sybil Attack

• A single physical node creates multiple identities to gain control in the network.

• In DHTs node ids are often used to form the topology of the network.
Eclipse Attack

- Routing table poisoning: incorrect routing table updates or join-leaves are used to choose the locations of the malicious nodes
- Routing tables can be separated to secure and optimized
- Protection against churn using random churn and central timed randomness service
- Cuckoo rule
  - routing from region-to-region
  - provably secure against adaptive adversarial join-leave attacks
Rendezvous Interconnect Security
Mechanisms

• Scope authorizes rendezvous networks to advertise themselves with cryptographic
certificates. The rendezvous network signs each advertisement in the interconnect.

• Each subhierarchy is controlled by a trusted 3rd party that can assign secure node
ids to interconnect nodes [Castro et al. 2002].
  • Each subhierarchy makes a contract with its parent to join the parent DHT.
  • Subhierarchies can collect statistics about the amount of resources used from other
hierarchies, but we are still working on accountability.

• DDoS
  • Game theoretic approach is not adequate because of botnets and possibly large
  externalities that are hard to measure.
  • Traffic limiting based on expected traffic distribution at each node of the DHT:
    – Daswani, N.: Denial-of-service (dos) attacks and commerce infrastructure in peer-to-
  • Rendezvous system is further protected by the underlying pub/sub routing that
    prevents unwanted traffic from the underlay.
Multirate Multicast Congestion Control with Caches

- Resource allocation view to congestion control: equilibrium between users and operators [Kelly97,98]:
  - maximize the aggregate utility that is a function of the receiving rate for each subscriber
- Popular flows should be given a larger share of bottleneck links.
- Single-rate multicast uses the rate of the slowest receiver.
- Solution: add chained congestion control loops at each branching point with large caches.
  - Alternatives: layered multicast, ROMA (uses erasure coding to eliminate need for large caches)
- Storing subscription state in branching nodes costs too.
Multirate Multicast Congestion Control with Caches

- On the subscriber side of the valley-free path, transit domains have the incentive to cache and branch.
  - Each subscriber has a weight for each subscription to signal its utility.
  - Each link and branching node memory has a price for its resources that is adjusted based on the elastic demand.
    - Proportional fairness, longer routes cost more
  - A (virtual) subflow should only pay for the links where it can receive at the rate of the maximum of the subflows.
- On the publisher side, the data source must compensate for the whole multicast tree → the data source must have control of the tree shape.

\[ x_r(t) = kr(x_r)(wr - x_r(t)qr(t) - cr(t)) \]

\[ p1(t) = f1(y1(t)) \]
Future Work

• We are currently working on implementing a simulation of the system and will report results in a coming paper.
  • efficiency, stability ..
  • analysis of the congestion control
• We will continue working on the security of the architecture.
• Open problems: multipath forwarding, reserved bandwidth and bursts etc.

• Thank you! Questions?