Publish/Subscribe Internetworking
From PSIRP to PURSUIT

George Xylomenos
Mobile Multimedia Laboratory, AUEB
xgeorge@aueb.gr

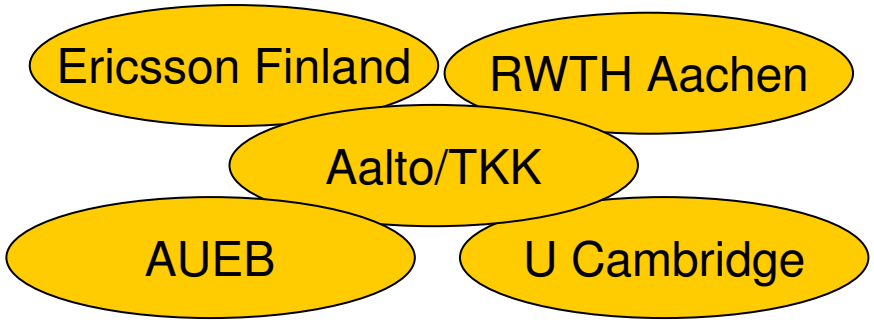
Motivation

- Why revisit internetworking?
  - No need to preach to the converted, but…
  - …shift emphasis from endpoints to content
  - …adapt the network to application patterns
- Why use publish/subscribe?
  - Inherently content centric
  - Natural model for many applications
  - Seems to solve some current problems
    - Especially the power of senders over receivers

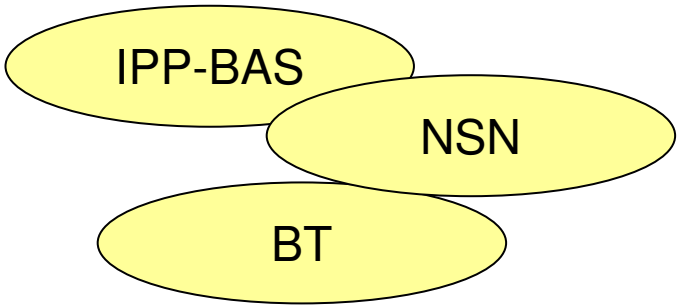
PSIRP and PURSUIT

- PSIRP: 2008-2010 (just ended)
  - Design and prototype the basic concepts
    - Rendezvous – Topology – Forwarding
  - Rough host and protocol implementations
  - Evaluation mostly by analysis and simulation
- PURSUIT: 2010-2013 (just started)
  - Explore lower and higher layers
  - Shift emphasis to inter-domain issues
  - More comprehensive evaluation

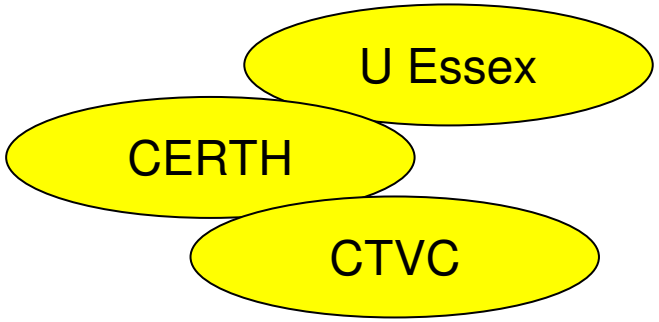
Project partners



PSIRP/PURSUIT



PSIRP only



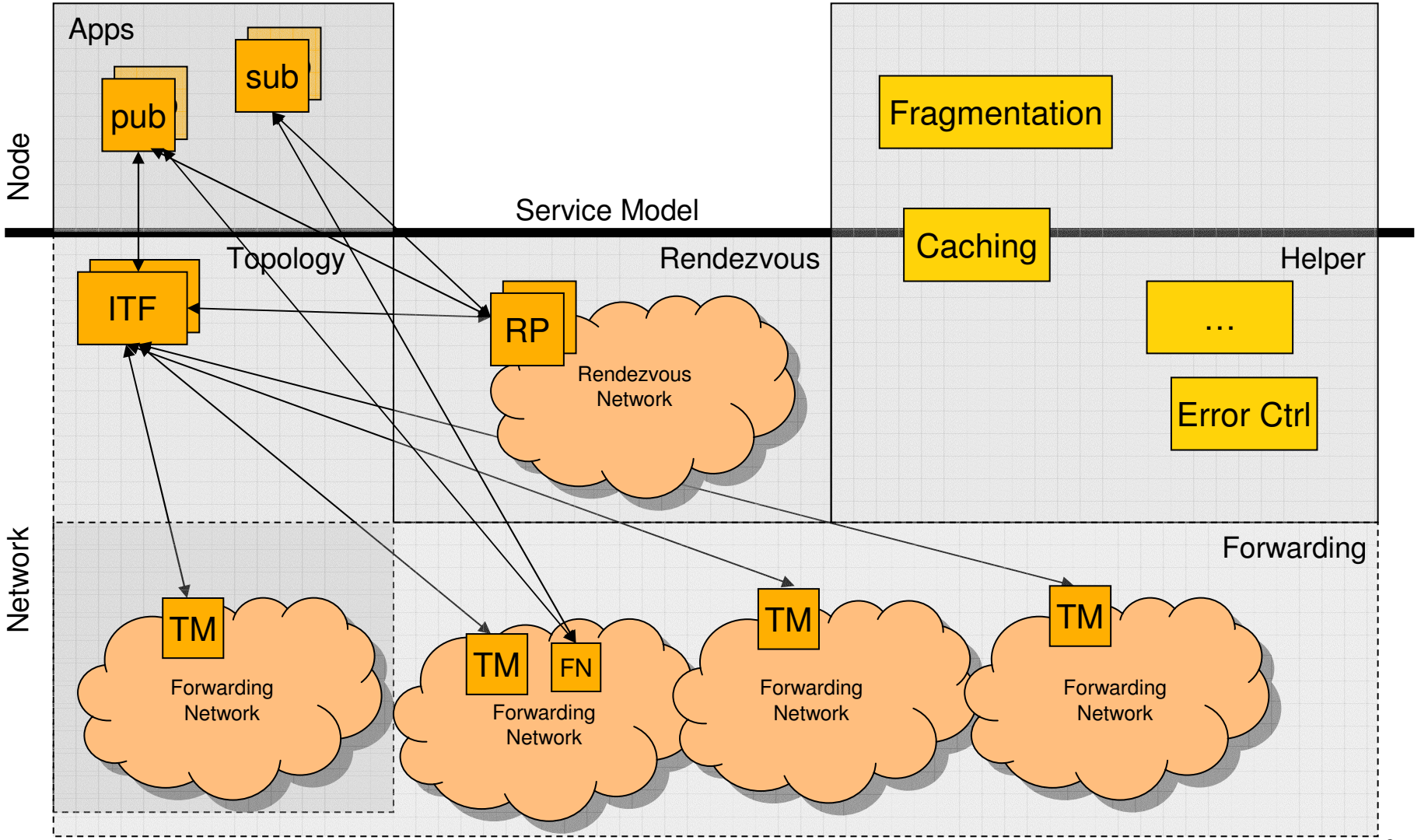
PURSUIT only

Working method

- Spiral development model
  - Design, prototype, evaluate, repeat
  - Worked surprisingly well so far
- PSIRP managed 2-3 iterations
  - Very exploratory initial phases
  - Three closely located partners
  - Loosely coupled modules (maybe too loosely!)
- PURSUIT will probably manage 2 iterations
  - We know more, but we aim for more!
  - More partners in development from the start

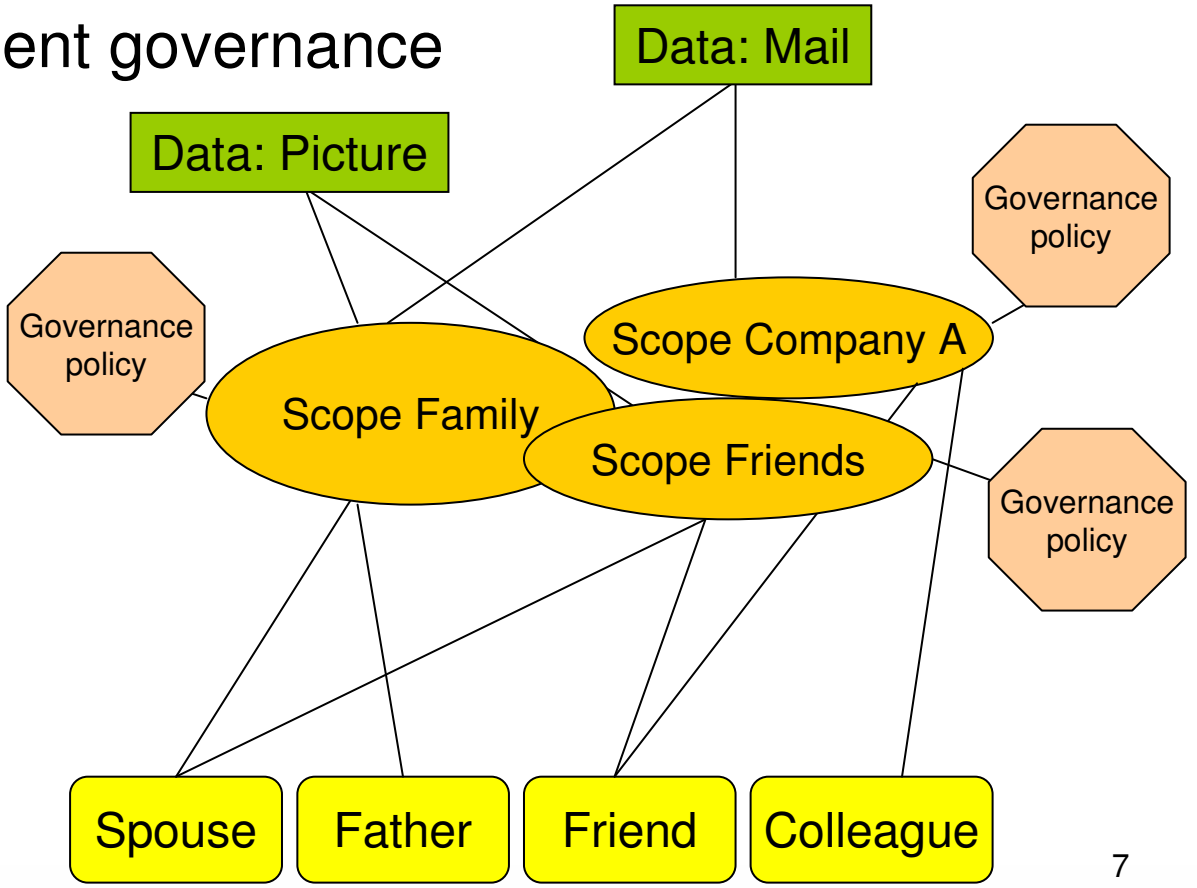
PSIRP
PUBLISH-SUBSCRIBE
INTERNET ROUTING
PARADIGM

High-level architecture



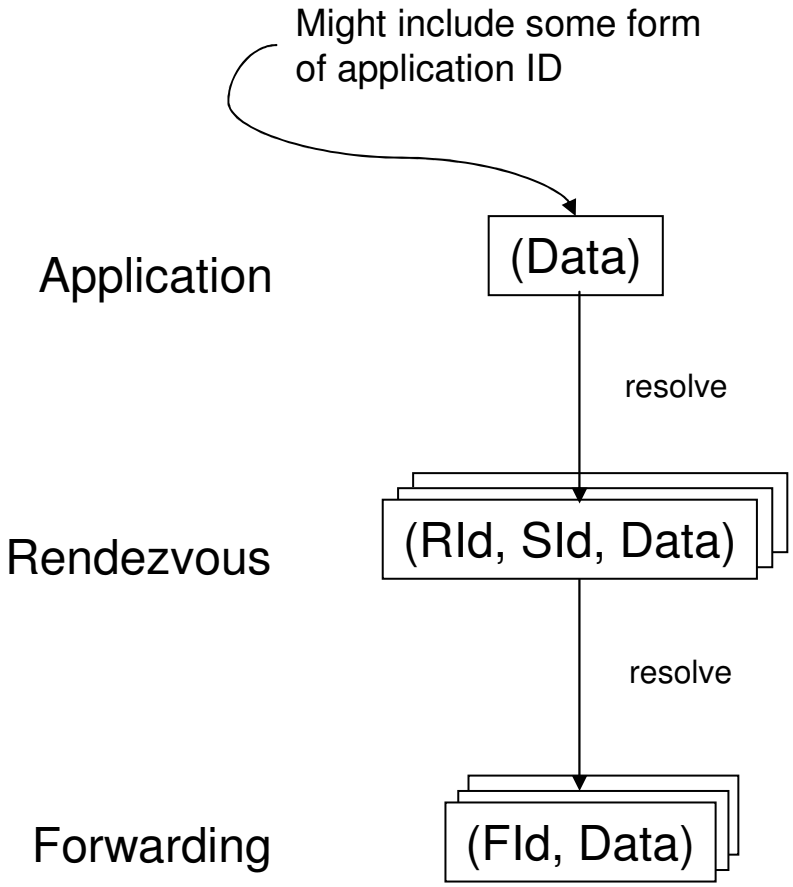
Information organization

- Information is published within (possibly many) scopes
  - Scopes are information collections
  - Scopes implement governance



PSIRP
PUBLISH-SUBSCRIBE
INTERNET ROUTING
PARADIGM

Information identifiers



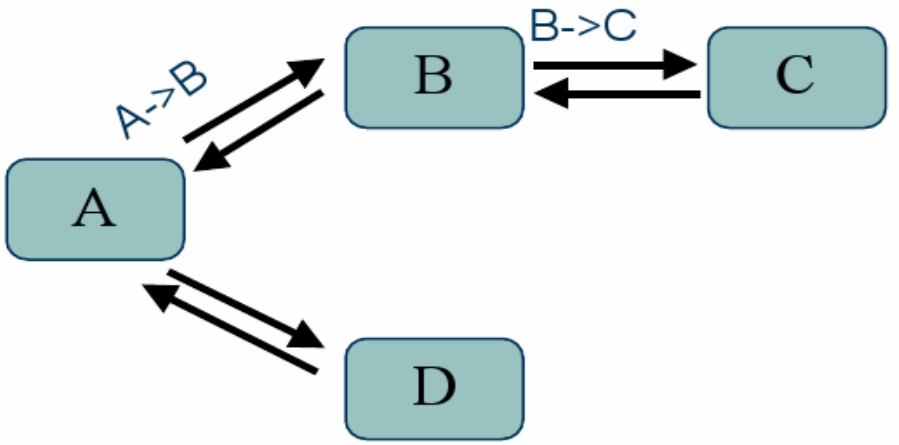
- Information is identified by
  - Scope ID (flat) and
  - Rendezvous ID (flat)
- Higher layer
  - Application IDs
    - Arbitrary
  - Resolution?
- Lower layer
  - Forwarding IDs
    - zFilters

Rendezvous

- Securely matches
  - entities (publishers and subscribers)
  - wishing to communicate (via publications)
  - on a certain topic (indicated by a RId)
  - inside a given scope (indicated by a SId)
- Locates rendezvous point(s) for a particular scope
  - Dedicated control plane (slow path)
- Two tier architecture
  - Individual rendezvous networks
  - Global rendezvous interconnect

Forwarding

- zFilter: in packet source route encoded as Bloom filter
  - Each link has a domain-local Link ID
  - Link tags of a path are combined
- Advantages
  - Fast forwarding
  - No local routing tables
  - Native multicast
- Disadvantages
  - Intradomain only
  - Extensions in PURSUIT

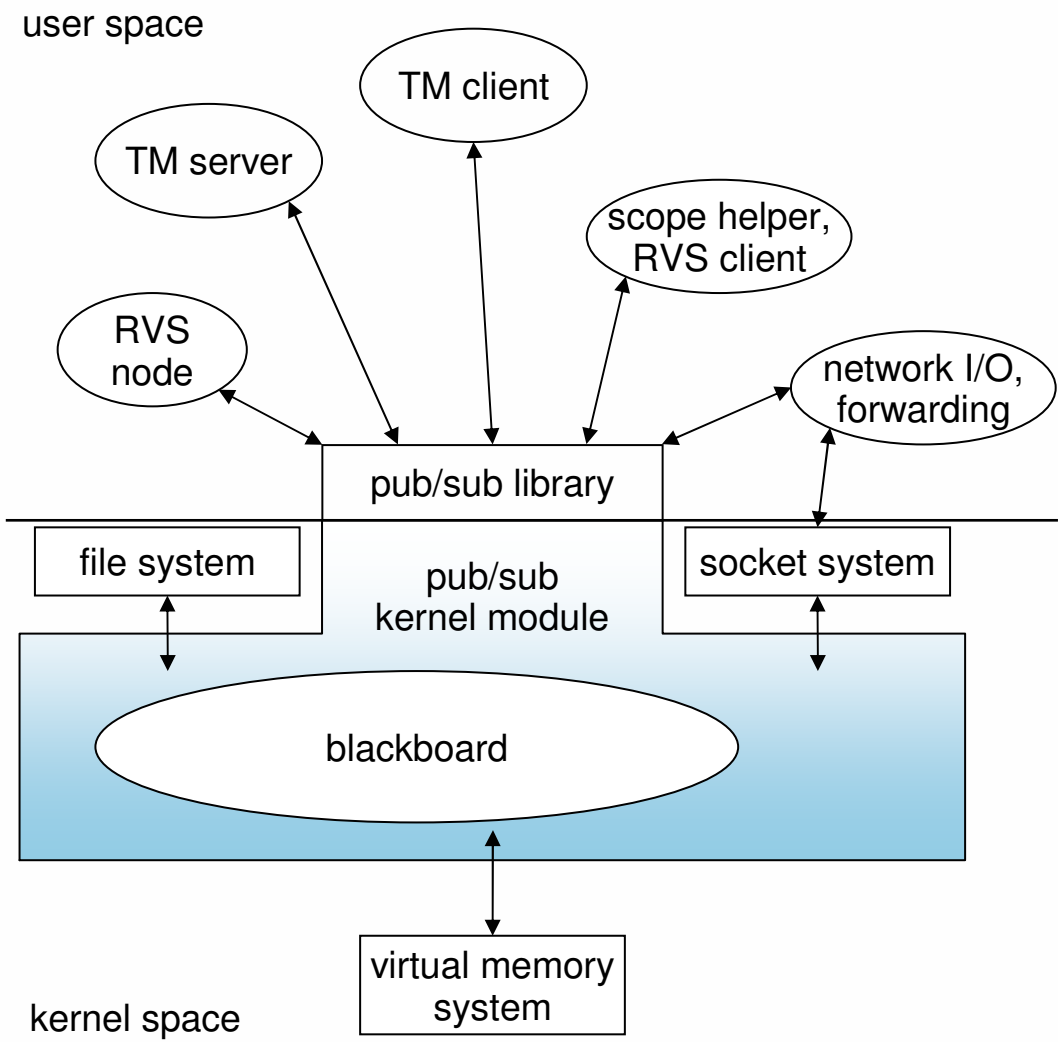


A->B	0	1	0	0	0	1	0	0	1
B->C	1	0	0	0	0	1	1	0	0
zF: A->B->C	1	1	0	0	0	1	1	0	1

Topology

- Intra-domain (PSIRP)
  - Topology manager(s) in each domain
  - OSPF like protocol distributes Link IDs
  - Shortest paths are encoded to zFilters
    - Easy to combine partial paths
- Inter-domain (PURSUIT)
  - Need to take routing policies into account
  - zFilters have limited capacity
    - Exploring label switching and label stacking

Native implementation



Native API

- Publications are simply memory areas
  - Sequences of pages
- Create publication
  - Allocate virtual memory objects
- Publish
  - Make content available to others
- Subscribe
  - Request and get content
- Register / Listen
  - Get notifications about publication events

Applications

- Some demo applications implemented
  - Firefox plugin, VoPSIRP, VidPSIRP, VLC tunnel
- How to get multiple Rids?
  - Assuming Google provides the first one…
- Algorithmic IDs: calculate sets of RIds
  - Good for streaming or segmentation
- BitTorrent-like: files with multiple RIds
  - Good for fixed documents
- Versioning: use the same RId repeatedly
  - Good for evolving documents

Overlay implementation

- Alternative implementation of PSIRP concepts
  - Implemented on top of IP
    - Pastry for key (ID) based routing
    - Scribe for rendezvous and multicast
  - More functionality for more overhead
  - Explored some higher layer ideas
- MultiCache: combined multicast & caching
  - Uses Scribe to serve flash crowds via multicast
  - Exploits Scribe state to keep track of caches
    - Caches serve later arrivals via unicast

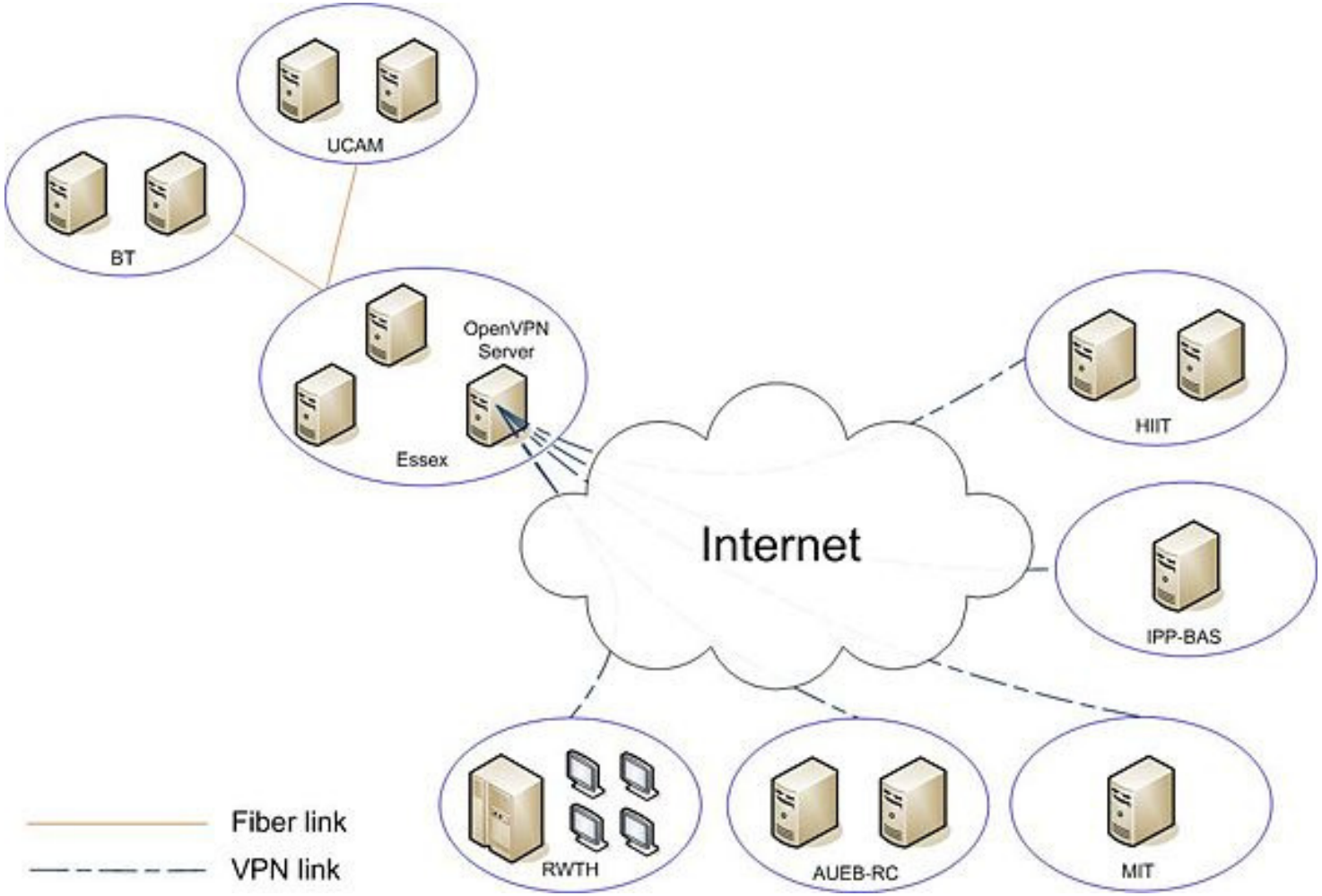
Qualitative evaluation

- Security evaluation
  - Red-team approach, many fundamental issues
    - What are the attacker types?
    - What can the attackers do?
  - Search for vulnerabilities
    - Many found in PSIRP, more expected in PURSUIT
- Socio-economic evaluation
  - Test architecture viability in an uncertain future
  - PSIRP: System dynamics approach
  - PURSUIT: Market based approach

Quantitative evaluation

- Modeling and simulation issues
  - Models for traffic, user behaviour, policies?
  - Proper dimensioning for simulations?
  - Proper simulation test bed?
    - NS-3 used for the native implementation
    - OMNeT++ used for the overlay implementation
- Real experiments
  - Isolated test beds at partner sites
  - Hard to use PlanetLab for native implementation
  - Used VPN based testbed between partners

Evaluation testbed



Dissemination

- Open source code releases
  - All PSIRP sources released to the public
    - BSD and MIT licenses
  - External code site and wiki
    - Sources and VM images
  - FreeBSD node and rendezvous implementations
  - NetFPGA forwarding implementation
- Project spin-offs
  - BitTorrent for OMNeT++ (for benchmarking)
  - Hierarchical Pastry (for global rendezvous)

Thorny issues

- Identified in PSIRP, to expore in PURSUIT
- Rendezvous semantics
  - PSIRP assumed 1-to-N (mostly)
  - Also need N-to-1 (at least for network attachment)
- Service model
  - Document model implemented in PSIRP
  - Can a (TV) channel model be retrofitted?
- Network specific issues
  - Optical, wireless, mobile optimizations
- Transport protocols
  - Especially for multicast (back to the future!)

Conclusions

- PSIRP was a very ambitious project
  - Publish/subscribe everywhere in the stack
  - Many issues tackled
    - Rendezvous, local forwarding and topology, simple API
  - Opened up even more
- PURSUIT is more and less ambitious
  - Pursues many new directions
    - Lower and higher layer issues
  - Exploits previous work
    - Code base and lots of mistakes