Yallcast Architecture Overview

Paul Francis

NTT PF Labs

francis@slab.ntt.co.jp

www.yallcast.com

“Distribution” Today: Two Parallel Tracks

- IP Multicast
  - Simple, automatic, standardized
  - Has problems, hasn’t reached “critical mass”
- Server-based
  - Broad functionality, almost everything server-based today
  - Application-specific, ad hoc, no standards, management-intensive

Yallcast Goal: Unify Both Tracks

- “Host”-based distribution tree
  - Tunneled over IP unicast (and multicast)
  - Buffering in hosts (or not)
- DNS name-based group addressing
- Dynamically self-configuring topologies

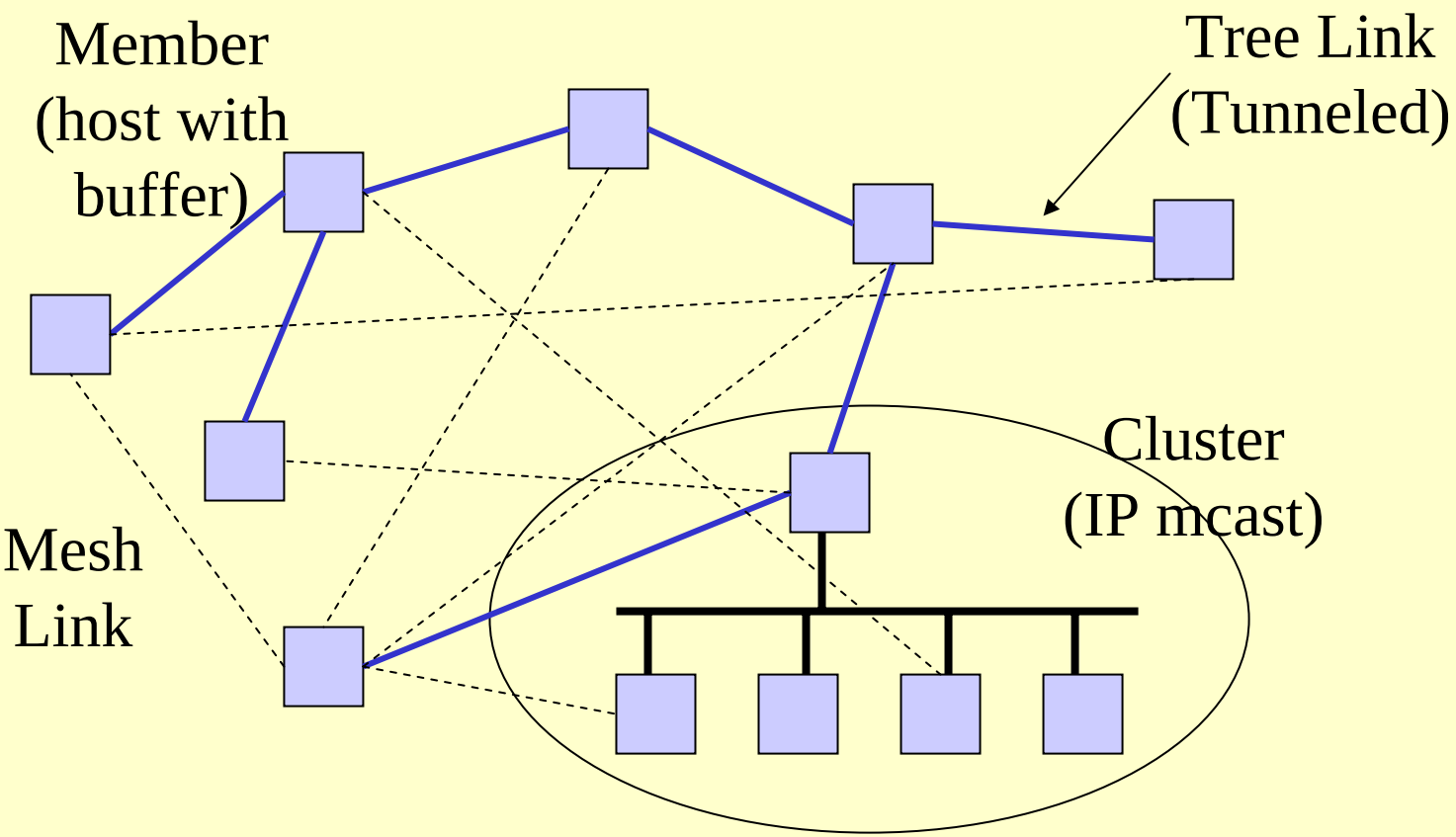
Status of Yallcast

- Basic algorithms worked out
  - Especially dynamic tree configuration
- Experimental implementation
  - Jan. 00 release target
- Many many open issues
- This talk is a call for participation
  - Certainly not a call for standardization

Yallcast Architecture Overview

- Rendezvous Nodes:
  - Bootstrap members into tree-mesh
- Member Nodes:
  - Dynamically configure into tree-mesh
  - Send, receive, and forward frames
- Group ID:
  - rendezvousName, treeName, [udpPort]

Yallcast Topologies



Yallcast Topologies

- Dynamically configured Tree and Mesh
- Both can carry content frames
- Tree Topology
  - Optimized for efficiency, but fragile
- Mesh Topology
  - Optimized for robustness, but inefficient

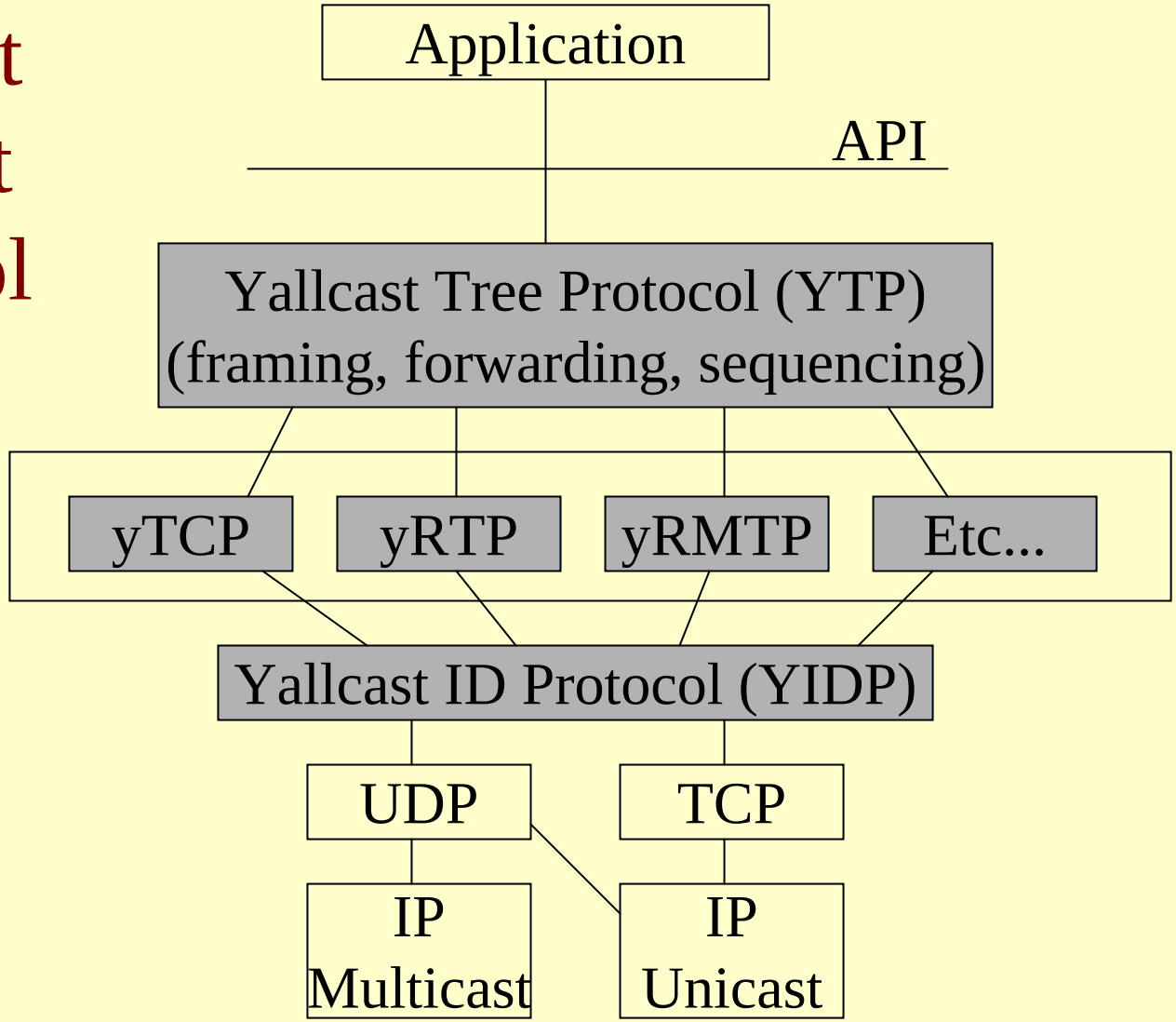
IP Multicast: Yallcast “Cluster”

- Group ID hashed into IP multicast addr
- IP Multicast tightly scoped
  - Currently to 1 hop
  - Admin scoping may be possible
- Cluster head member dynamically elected
  - Joins rest of tree-mesh
  - Other members send/receive via IP multicast

Reduced Role of IP Multicast

- IP Multicast always runs under yallcast
- IP Multicast no longer expected to have global scope

Yallcast Content Protocol Stack



Member Identification

- Based only on:
  - Member domain name
  - Yallcast port (32-bit locally unique number)
- Not based on IP or UDP/TCP port
- Member “how to reach” information carried separately
  - IP addresses (including NAT box), ports, etc.

Yallcast Content Protocols

- Application frame-based
- Per-source 64-bit byte sequencing
  - Frame can be forwarded over tree or mesh
- Tag-based headers (hop by hop)
  - Frame source id --> 16-bit tag
  - HxH source id, HxH dest id, group id --> 64-bit tag

Comparison to IP Multicast

- ↑ Routing table scalability
- ↑ Group ID (address) assignment
- ↑ End-to-end Reliability
- ↑ Congestion Control

- ↓ Proximity discovery
- ↓ Delivery efficiency (for non-reliable)

Trickier Comparisons

- Evolutionary Path
  - Don’t need any infrastructure in advance
  - Just bundle with app
  - Add infrastructure as needed
- Buffering
  - Hosts have lots of buffer---async distribution
  - But introduces new coordination problems

Rendezvous Node's Algorithm

- rendezvousName, treeName, [udpPort]
- Listen on udpPort
- Keep list of (some or all) group members
- Tell new members of existing members, group parameters (buffer size, security, etc.)
- Partition detection (detect multiple roots)
- Convenient place for other services.

Member Node's Algorithm

- Check local IP multicast for other members
  - If exist, join local cluster
  - May optionally contact Rendezvous
- If none, contact Rendezvous
  - Learn of existing members
- Run Yallcast Tree Management Protocol (YTMP) with existing members

Yallcast Project Next Steps

- Build real applications over yallcast
- Develop yallcast under real applications
- Work towards open-source environment
- Early standardization neither necessary nor appropriate
  - Standardize when ready for OS and proxy-server deployment