ALM API for Topology Management and Network Layer Transparent Multimedia Transport

<draft-lim-irtf-sam-alm-api-00.txt >
71ST IETF SAM-RG MEETING, PHILADELPHIA.

Lim Boon Ping
Ettikan K.K

Requirement

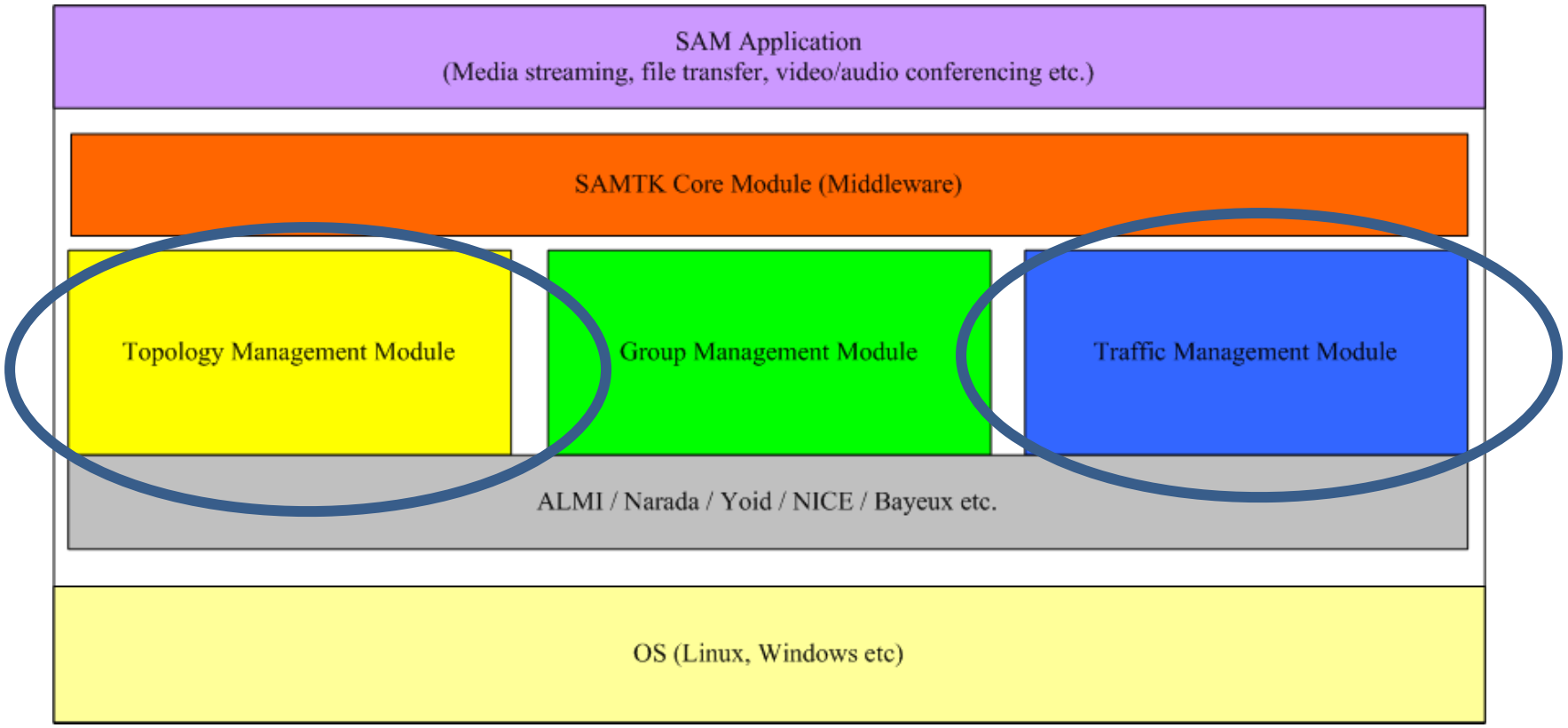
- Lack of common set of wrapper API leads to redundant application layer development, thus diluting effort for underlying ALM protocol enhancement and integration on a common application platform.

- Core functions of ALM topology management
  - ALM topology construction (at initial stage),
  - ALM topology re-construction (upon membership change),
  - ALM topology refinement (upon network condition or performance metrics change),
  - ALM topology distribution (for centralized approach),
  - ALM forwarding table lookup (upon data delivery/relay), and
  - Content distribution based on ALM topology.

Goal

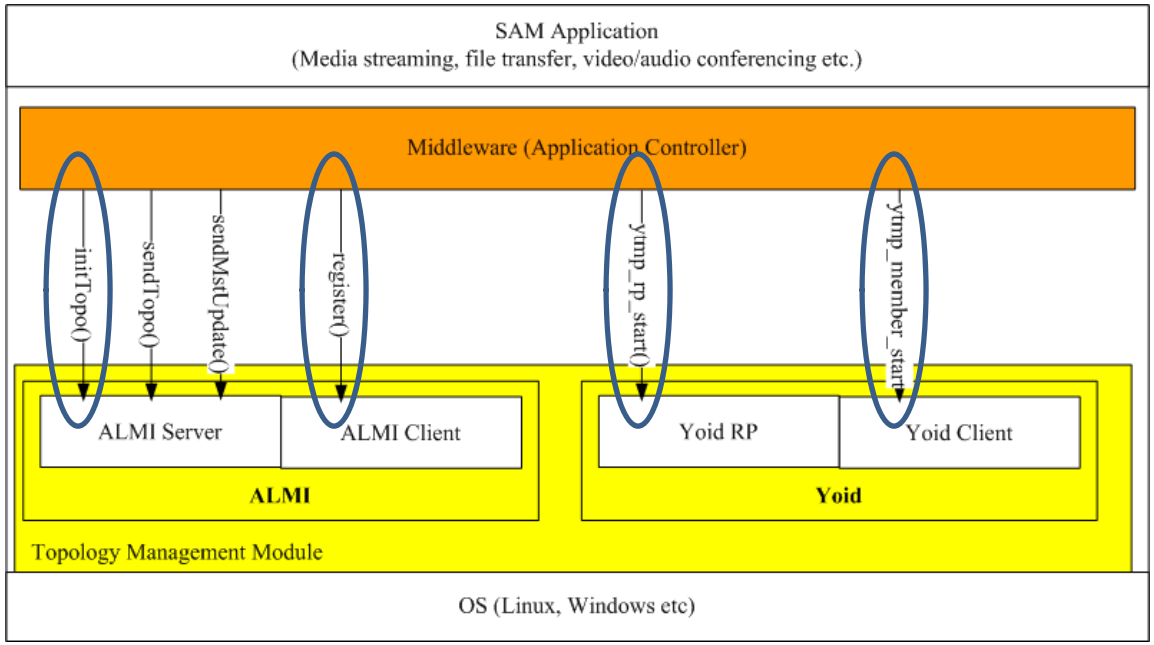
- SAMRG
  - Group Management
  - Traffic Management
- Proposal
  - Wrapper API for Topology management
  - Wrapper API for Traffic Management (flexible protocols selection)

Middleware Architecture



Needs for ALM Topology Management Wrapper API

- **Limitation of Protocol-specific API**
  - ALM protocols export different set of protocol-specific APIs despite providing common services.
  - ALM middleware is restricted to access only one ALM protocol by interfacing with protocol-specific APIs.
  - ALM middleware looses the flexibility to leverage on or access different ALM protocols based on the application needs within the same architecture.

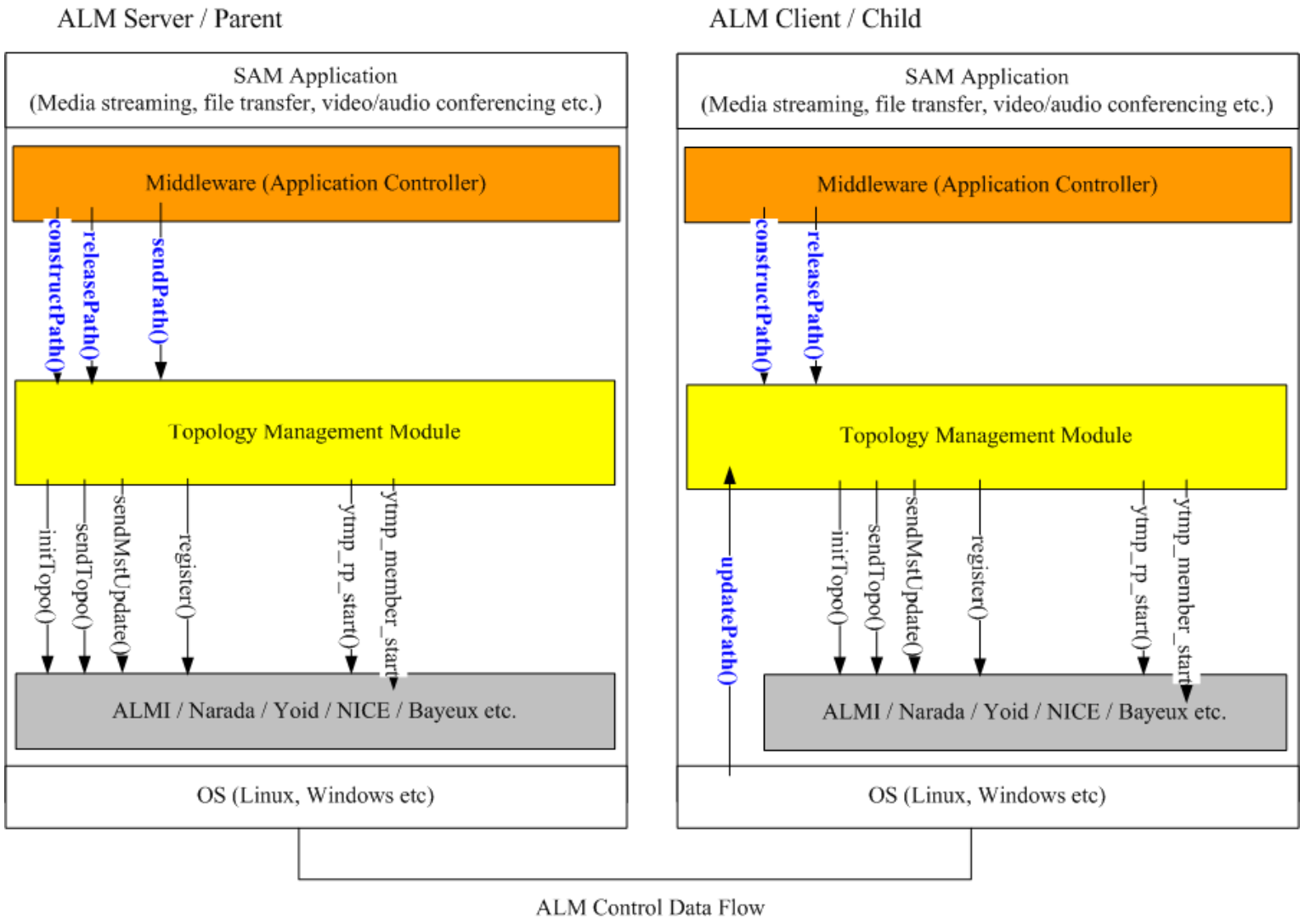


Needs for ALM Topology Management Wrapper API

- **Lack of complete coverage of different ALM APIs**
  - Centralized (eg. ALMI etc.) vs. distributed topology (eg. narada, yoid, bayeux etc.).
  - Overlay-based common API
    - RelayCast - *MakeTree(), Find(), SendDat(), RecvDat()*
    - Dabek et al. - *forward()*, *join()*, *leave()*, *local_lookup()*, *multicast()* and *anycast()*
  - Client-server centralized-based common API ?
    - ALM server
      - Initialization for receiving membership information, metrics information…
      - construct/update/distribute topology (a.k.a forwarding table)
    - ALM client
      - Receive/update forwarding table from ALM server,
      - perform forwarding table lookup to identify next destination ALM node for content relay.

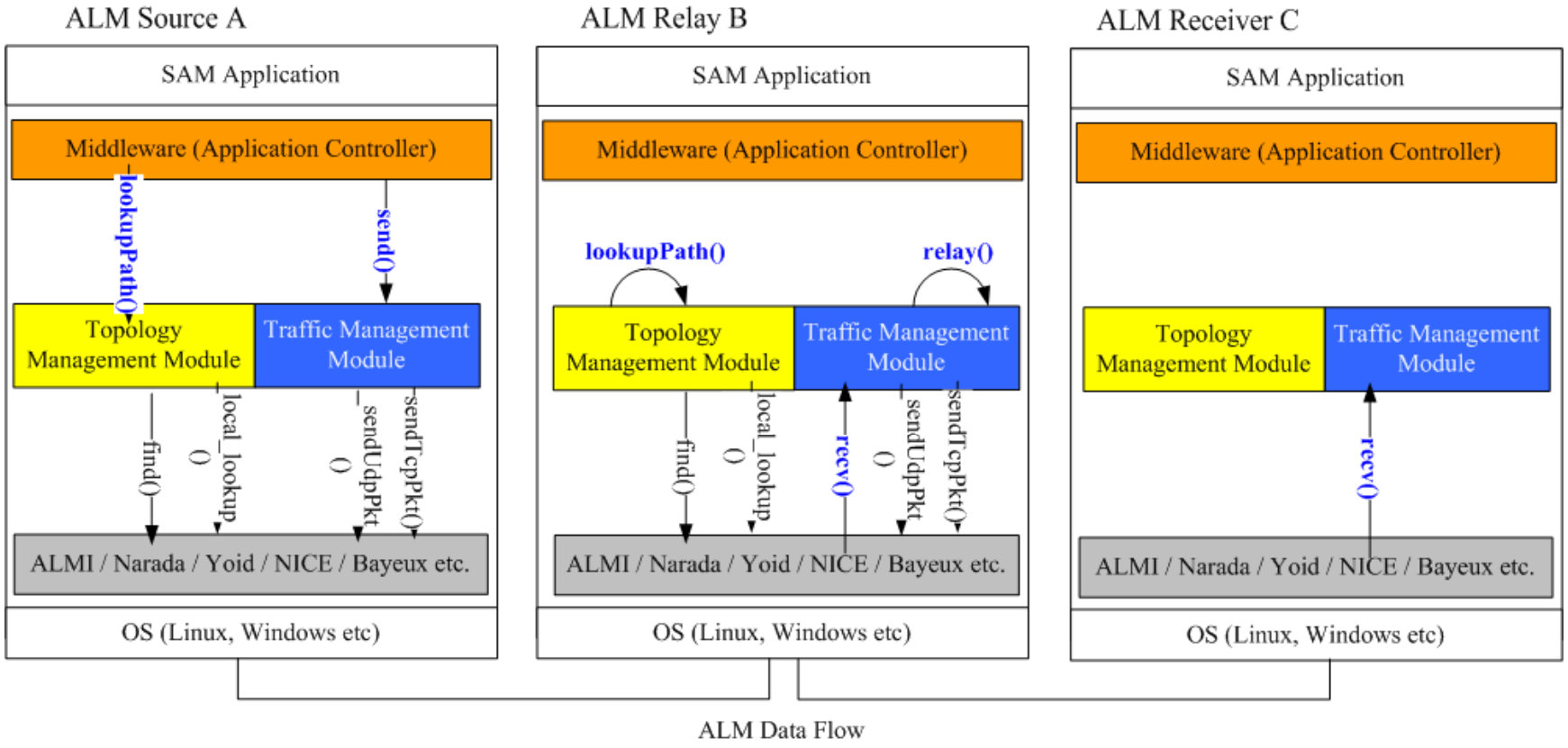
Usecase Example

Middleware API usage for ALM Tree Construction Approach



Usecase Example

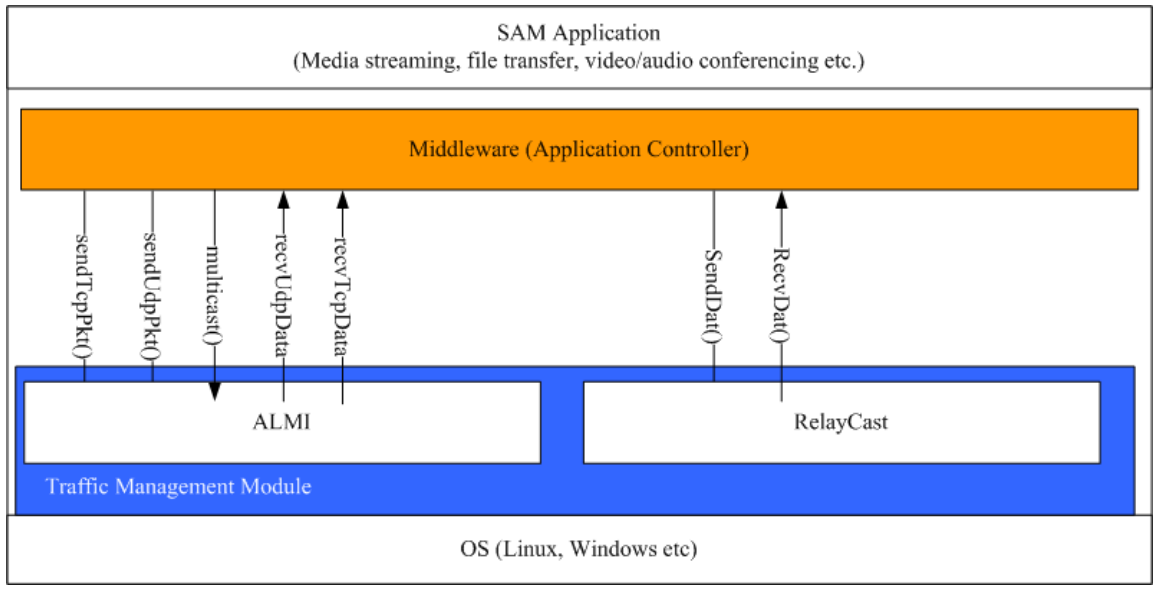
Middleware API usage for ALM Content Distribution / Relay



Backup

Needs For Network Layer Transparent Wrapper API

- **Lack of multimedia network/transport layer selection**
  - ALM protocols define different set of API for content distribution. Neither provides complete selection of IP protocol (v4, v6), transport protocol (tcp/udp/rtp) or multicasting type (IP multicast, multi-unicast, xcast) specifically from the ALM middleware.
  - Flexibility to select the network and transport layer protocols, and the underlying Traffic Management module to switch ***dynamically*** based on network capability and application need



Middleware API For Forwarding Path Construction & Distribution

- **`constructPath`**`(const int actionMode, struct AlmPathLst* almPaths, struct AlmMetricsLst* almMetrics)`
  - actionMode – BUILDTREE, JOIN
  - almPaths – (output) forwarding table / selected parent path information
  - almMetrics – (optional input) provides metrics information for path construction from external metrics collection/providing module.

- **`releasePath`**`(const int actionMode)`

- **`sendPath`**`(const int transportMode, const in sendMode, struct AlmPathLst* almPaths)`
  - transportMode – mode of sending table: IPV4, IPV6, AUTO etc.
  - sendMode – NODESPECIFIED, FULL

- **`updatePath`**`(struct AlmPathLst* almPaths)`

Middleware API for Content Transmission / Receiving / Relay / Path Lookup

- **send**(const int transportMode, struct AlmData* data)
  - transportMode – ALMCASTV4UDP, ALMCASTV4TCP, ALMCASTV6UDP, ALMCASTV6TCP, IPMULTICAST, AUTO, variant of XCAST6 implementation etc.

- **recv**(struct AlmData* data)

- **relay**(const int transportMode, struct AlmData* data)

- **lookupPath** (uint32_t key, struct AlmDistLst* almDist)
  - Key – primary key to lookup
  - almDist – list of next destination receivers

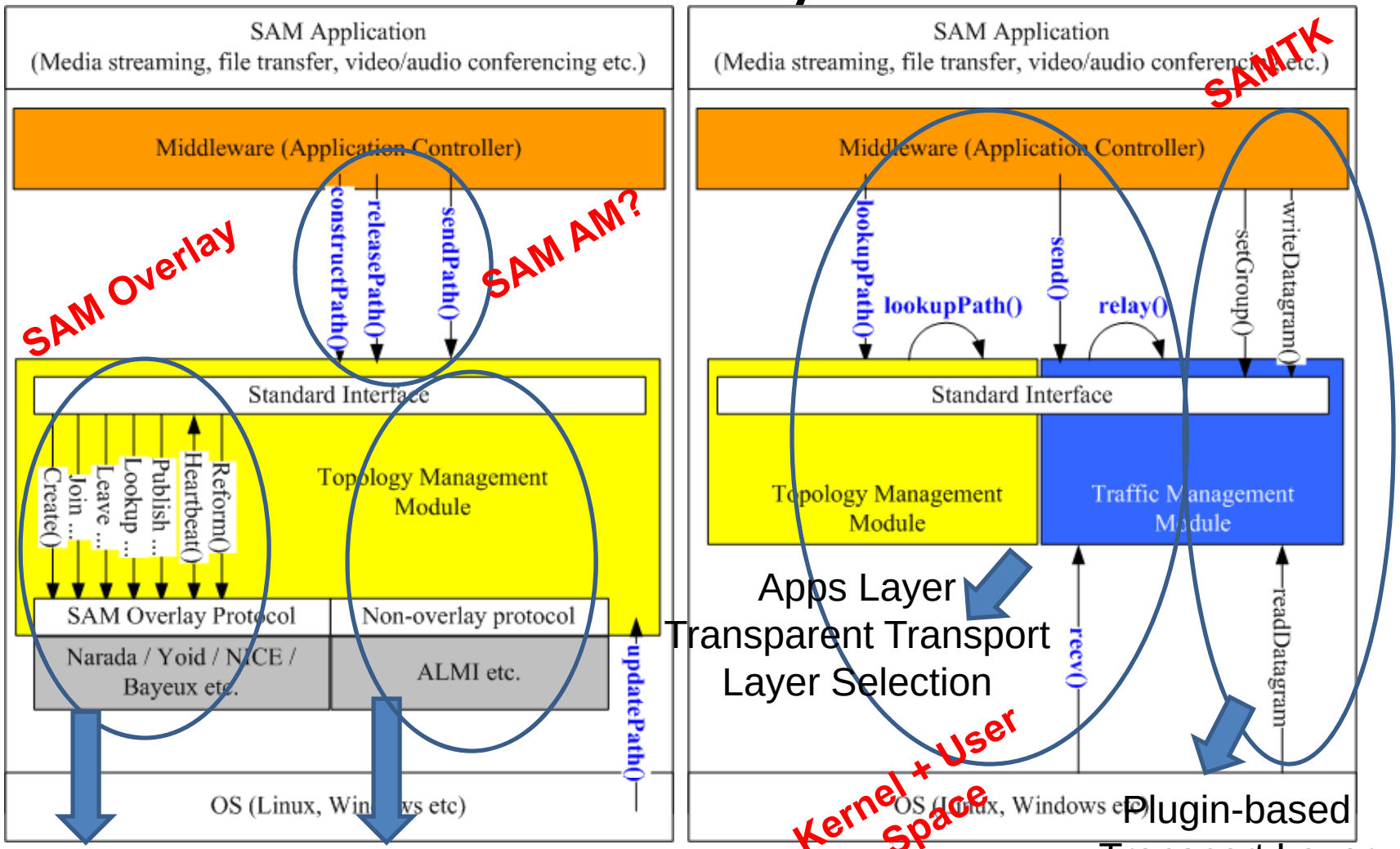
Objective

- To understand & identify common requirement for SAM framework development & API definition, covering aspects of
  - Topology management (Multi algorithm support)
    - Application Layer Multicast
    - Overlay Multicast
  - Traffic management (Multiple protocol support)
    - Native multicast
    - Xcast
    - Proprietary unicast (eg. ALMcast)
    - etc.
  - Group membership management

Comparison of SAMTK / ALM API / SAM Overlay Protocol

Description	SAMTK	ALM API	SAM Overlay Protocol
Goal	Define the **group management and traffic management API** with topology management module is anticipated to be managed by ALM/OM plugin developer.	Define the **topology management** and **network layer transparent middleware wrapper API** for ALM forwarding table construction, distribution and multimedia transport over IPv4/IPv6 for unicast and xcast.	Define the **overlay API** and messaging needed to • support the multicast tree operations between ALM/NM region via AMT-GW. (Propose P2PP message for SAM to ensure SAM framework to be compatible with P2P SIP). • ensure overlay agnostic API so that different overlay algorithms can interoperate in a single SAM (Scalable Adaptive Multicast) session.
API consideration for Transport protocol	Define SAMSocket with underlined protocol can be called through standard plugin interface (xcast, multicast, ipv6, alm)	Transport layer transparent via selection of underlying transport protocol. May work as another higher level API on top of SAMTK standard plugin interface. (necessary?)	Generic multicast message. Note: Use AMT (Automatic IP Multicast Without Explicit Tunnels) to connect peers in ALM (Application Layer Multicast) regions with peers in native multicast regions.
API consideration for Topology Management		Centralized/distributed non-overlay-based ALM.	Overlay multicast
API consideration for Group Management	Define list of group management related API		

Comparison of SAMTK / ALM API / SAM Overlay Protocol



Comparison of SAMTK / ALM API / SAM Overlay Protocol

Proposed API	SAMTK	ALM API	SAM Overlay Protocol

Topology Management API		constructPath(const int actionMode, struct AlmPathLst* almPaths, struct AlmMetricsLst* almMetrics);
		
		releasePath(const int actionMode);
		
		sendPath(const int transportMode, const in sendMode, struct AlmPathLst* almPaths);
		
		updatePath(struct AlmPathLst* almPaths)
			Create(PeerId, SessionKey, GroupId, Options) ;
			JoinAccept(ParentPeerId, ChildPeerId, GroupId, Options);
			JoinConfirm(ChildPeerId, ParentPeerId, GroupId, Options);
			JoinDecline(PeerId, ParentPeerId, GroupId);
			JoinViaAMTGateway(PeerId, AMT-GW, GroupId, Options);
			LeaveViaAMTGateway(PeerId, AdjacentPeerId, AMT-GW, GroupId, Options);
			Heartbeat(PeerId1, PeerId2, GroupId);
			PublishAMTGateway(PeerId, Key, Region, Options);
			LookupAMTGateway(PeerId, Key);
			PublishPeerNMInfo(PeerId, Key, Options);
			LookupPeerNMInfo(PeerId, Key);

Comparison of SAMTK / ALM API / SAM Overlay Protocol

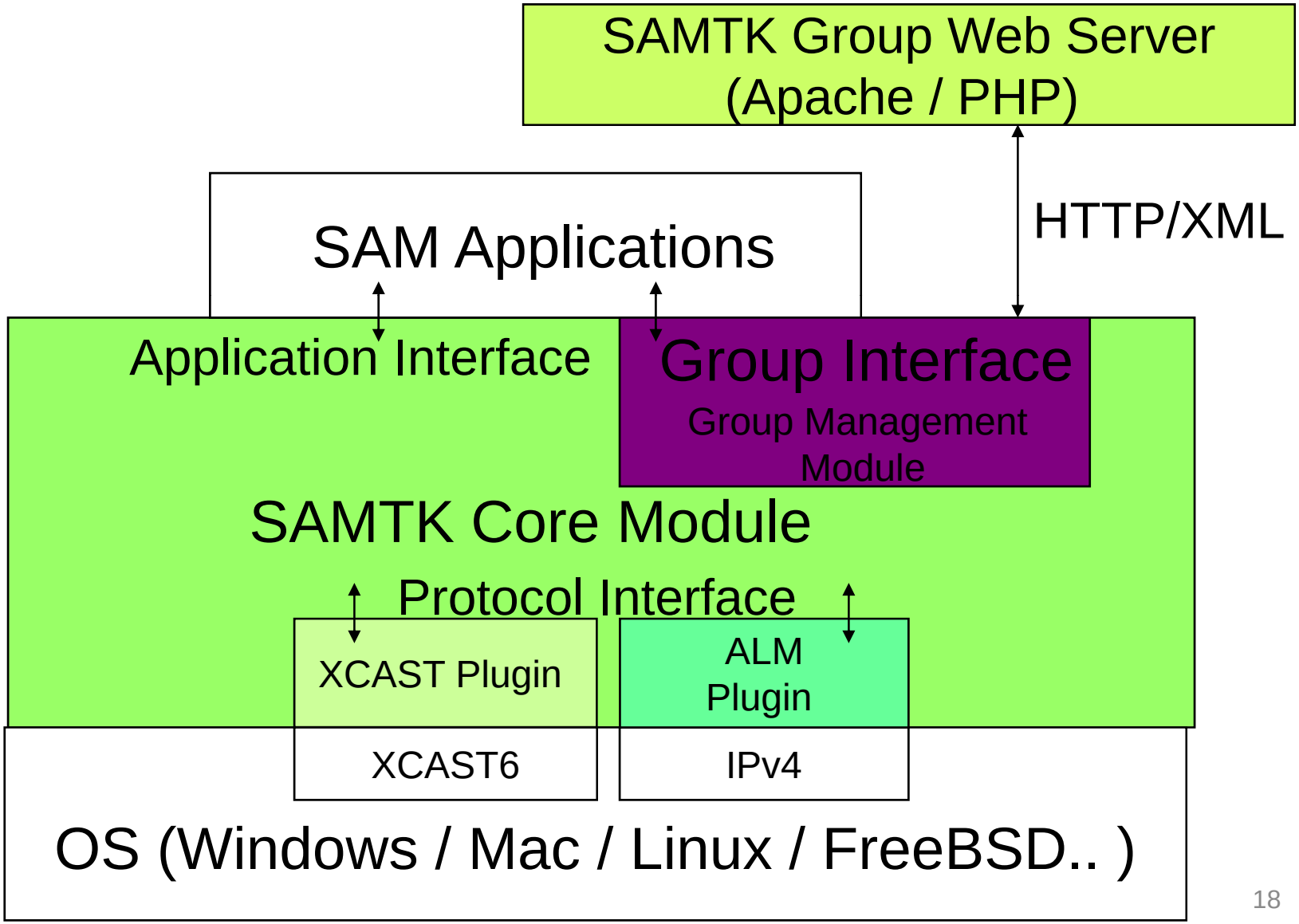
Proposed API	SAMTK	ALM API	SAM Overlay Protocol

Traffic Management	SAMSendSocket
	SAMReceiveSocket
	
	setGroup(GroupAddress);
	writeDatagram(char *, int, GroupAddress);
	readDatagram(char *, int, HostAddress);
	bool hasPendingDatagrams();
		send(const int transportMode, struct AlmData* data);
		
		recv(struct AlmData* data) ;
		
		relay(const int transportMode, struct AlmData* data)
		
		lookupPath (uint32_t key, struct AlmDistLst* almDist)
			multicastMsg (groupID, msg);

Group Management	getSAMGroupMemberList(GroupURI);
	getSAMGroupMember(MemebrURI);
	getSAMGroupInfo(GroupURI);
	getSAMGroupAddress(GroupURI);
	addGroup(newGroupURI, path);
	deleteGroup(GroupURI);
	addMember(GroupURI);
	joinGroup(GroupURI, properties);		

	deleteMember(MemberURI);
	setProperty(MemberURI, Key, Value);
	deleteProperty(MemberURI, Key, Value) ;		

SAMTK Architecture



DMMP vs. SAM Overlay Protocol

Description	DMMP	SAM Overlay Protocol
Goal	A new dynamic mesh-based OM protocol framework, with protocol details for data and control plane: •Session initialization & •Super node selection •Member join/leave •Refresh (heartbeat) information •Data delivery control •Failure recovery •Self-improvement (optimization)	Define the **overlay API** and messaging needed to • support the multicast tree operations between ALM/NM region via AMT-GW. (Propose P2PP message for SAM to ensure SAM framework to be compatible with P2P SIP). • ensure overlay agnostic API so that different overlay algorithms can interoperate in a single SAM (Scalable Adaptive Multicast) session.
Member join/leave	Member join/leave in dynamic OM-based cluster (super node-based cluster).	Member join/leave in ALM with additional consideration to NM region over AMT-GW.
Node selection	Details of super node selection, optimization (via node promotion), number of node per cluster/level are discussed.	No specific discussion on node selection.
Keep alive information	Refresh message is sent to peer periodically. If peer does not receive refresh message on time, it shall send PROBE message before notifying others.	Heartbeat message is sent to peer periodically. No probe is further sent