UNITE: An Architecture for Lightweight Signalling in ATM Networks

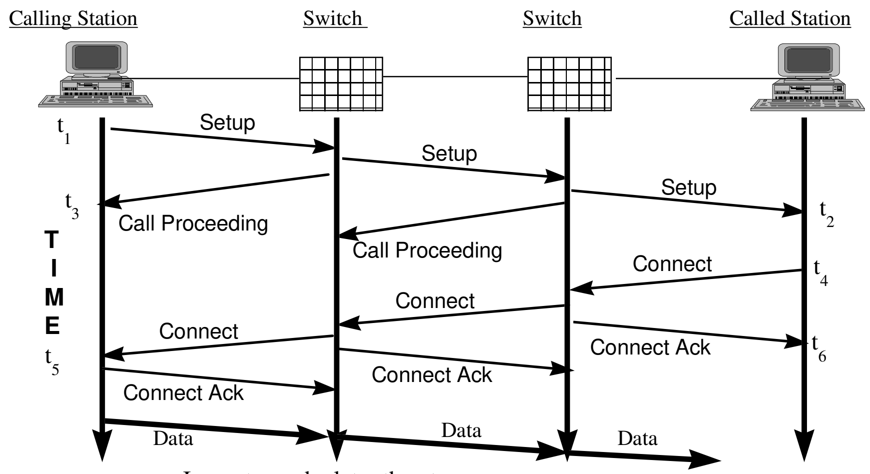
Gísli Hjálmtýsson and K. K. Ramakrishnan
Networking and Distributed Systems Research
AT&T Labs - Research

April 17, 1997

AT&T

1

UNI Call Setup Procedure



Issue: t_6 can be later than t_5.
Calling station's transmit cell can arrive before Connect Ack arrives at the Called Station.

AT&T

2

Some fundamental drawbacks of ATM signalling

- Inextricable tie-up between connectivity and comprehensive description of connection characteristics
  - result: poor signalling performance, even for simple non-QoS connection
    - » latency to setup a connection; throughput of # connections setup
- Inflexible models for who chooses to request QoS: sender only
- Changing parameters on existing connection is exception mechanism
  - even for same QoS; not adaptive enough for applications
- All participants on connection have same QoS parameters
- Strict sequential processing of quality & connectivity requirements
  - doesn't exploit parallelism possible with processing at multiple switches
- Poor at failure recovery - re-routing individual connection hard

➪ (also recovery from mistakes in admission control: break_then_make model)

AT&T

3

UNITE Goals: UNI ~~LI~~ TE

- Separate connectivity from defining/establishing QoS of connection
  - setting up basic connectivity is as fast as possible
- Overcome the speed of light - (requires per VC queueing)
  - start sending before completing round-trip connection establishment
    - » send after 1 hop-by-hop round-trip + ε
  - single cell call setup: no SAR, fixed length fields;=> hardware implementation

- QoS establishment & renegotiation are similar
  - enabling simple ways to change parameters for flows
- Parallel processing of QoS requests
  - even QoS establishment can be "quick"
- QoS setup initiated by source *or* destination.

➪ QoS setup failures may be more "benign" - retransmit periodically?

AT&T

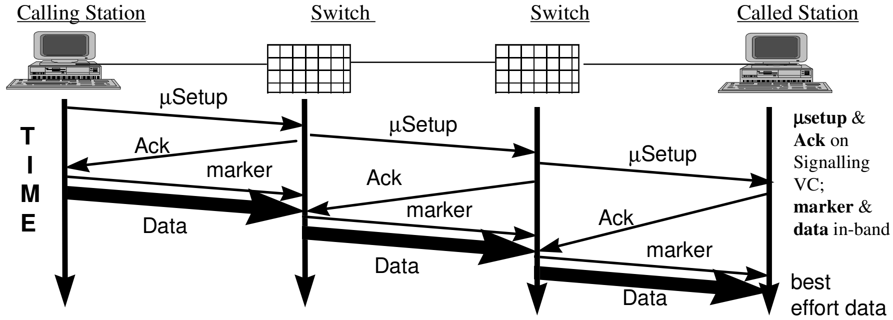
4

UNITE: Summary of Mechanism

- Uses a separate initial mechanism for connection setup
  - minimal function to have *best-effort* connection source-to-destination
    - » over path that satisfies some broad requirements
  - progresses on hop-by-hop
    - » data may be transmitted by upstream node immediately after hop-negotiation succeeds (uses per VC buffering)
    - » data transmitted is buffered until connection is established on next hop
- Initial setup includes QoS class identification for routing call setup
- Use separate QoS specification & negotiation
  - performed in-band, on established best-effort VC
  - QoS flow can begin after QoS negotiation completes
    - » QoS initiator can be source (sender) or destination (receiver)

AT&T

Micro-Setup: Best-Effort Connectivity

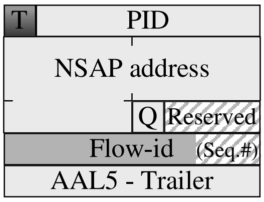


- route lookup for µsetup; obtain port to which µsetup is forwarded
- allocate VC from VC address space for upstream link & buffering
- forward µsetup downstream based on "best path"
- return VC id to upstream node.

AT&T

✚ (we have chosen to provide reliable delivery within signalling, instead of layering on top of "SSCOP")

Microsetup Cell Structure



- Destination Address (20 bytes) NSAP address
- Protocol ID (6 bytes) - dest. network layer entity
- QoS Class (1 byte) - for minimal QoS sensitive routing
- Type (1 byte) - type of signaling cell
- Flow ID (8 bytes) - unique (end-to-end) flow-id identifying the micro-setup from source.
  - the source host “Ethernet” address (6 bytes)
  - sequence number unique at source(2 bytes)
- AAL5 Trailer (8 bytes) - std. including the CRC and length.
- Reserved (4 bytes) - for future use
- Fixed cell format, no TLV encoding: for hardware implementation
  - ✚ single cell: no problems resequencing
  - ✚ easy to do distributed call-setup on per-port basis.

Micro-Setup Details

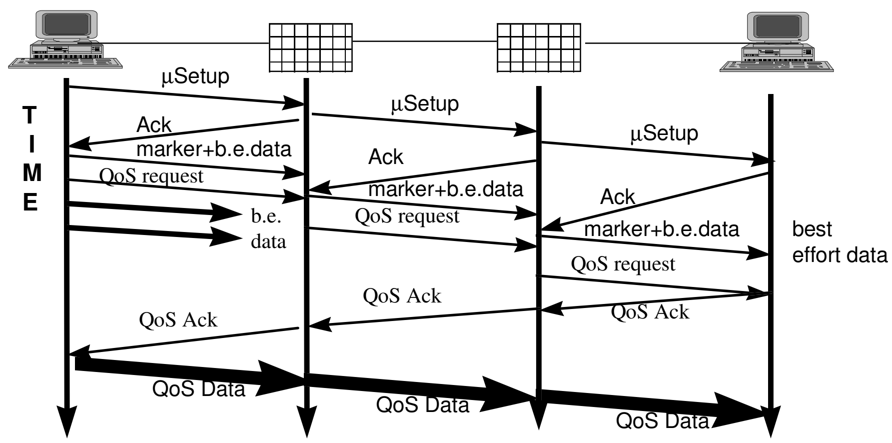
- ◆ µsetup and Ack. sent on well-defined “signaling” VC.
- ◆ Ack. includes: Flow ID (8 bytes); VC id returned (3 bytes); reason code for failures (1 byte).
- ◆ µsetup includes a QoS class “hint”
  - allows QoS sensitivity in routing
    - » reduce P(repeated failure of QoS request), on best-effort connection
      - ➪ if µsetup performed cognizant of QoS class: choice of path that has greatest likelihood of satisfying QoS requirements as suggested by “hint”.
    - » typically a small number of possible alternative routes.
    - » success for QoS class request often => detailed QoS request success
- ◆ Initial QoS classes: Best-Effort; Delay Sensitive; High Bandwidth
  - include a “nibble” for logarithmic encoding of parameter

QoS Flows

- QoS setup and Acknowledgment done in-band
  - best-effort or class adequately specified by "hint" requires no QoS setup
- In-band QoS Signaling allows:
  - same mechanism: QoS setup & renegotiation
  - request issued by sender *or* receiver
  - potential to exploit parallelism in processing of QoS/admission control at different switches
  - complexity for those flows requiring it; incorporation of *policy*.
- QoS message sent in-band:
  - identified as "Resource Management" cells;
  - AAL5 framing;
  - encoded in existing ways (e.g., UNI/Q.2931 or even RSVP)

9

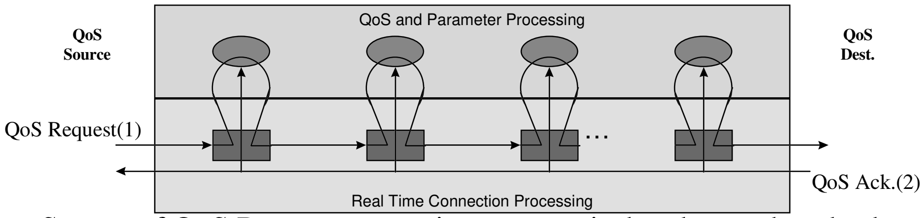
Establishing a QoS Flow



- Additional messages needed to establish QoS flows
  - worthwhile if (a) exception *or* (b) exploit parallelism in QoS processing

10

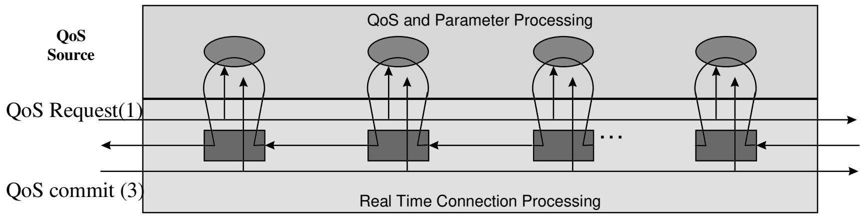
Exploiting Parallelism in QoS Processing



- Source of QoS Request transmits message in-band, goes hop-by-hop
- Admission control and downward negotiation of QoS parameters also hop-by-hop
- Take advantage of multicasting in switch for QoS Ack: some speedup
- Data continues to flow on best-effort basis until QoS negotiation completes & Ack. returns to QoS source

AT&T

Exploiting Parallelism in QoS Processing (contd..)



- QoS Request is multicast to switch: enables Admission Control functions to proceed in parallel
- QoS Ack. goes hop-by-hop to achieve downward negotiation
- Data using QoS can flow upon receipt of Ack
- A 3rd pass of QoS commit needed to "announce" final resource commitments to network (flow unaffected)
  - ➪ Mechanism helps if CAC functions are complex and parallelism helps achieve speedup

AT&T

Exploiting Parallelism in QoS Processing (contd..)

- A 3rd option (variant of previous): very much like RSVP signaling
- QoS request: an "announcement" to network of requirements
  - take advantage of multicasting within switch for "parallel processing"
  - failure to provide requested QoS/parameters
    - » indication by QoS processor using in-band "Reject"
- Source or destination may transmit QoS request
  - collisions resolved gracefully, with policy decisions left to end-systems
  - if QoS request is identical at a switch, merge request at switch
    - » source sends "Ack"
  - mismatch of QoS request: switch forwards request unprocessed to end-system
    - » end-systems decide who re-issues QoS request

AT&T

13

RELEASE

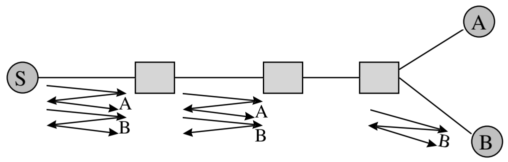
- Difficulty: don't know the addresses of both end-systems
- Cannot always count on availability of VCid to release VC
  - may have been re-used
- Two methods of performing RELEASE:
  - "voluntary release" by end systems, taking advantage of existing VC
    - » in-band, single cell
  - "involuntary release" - any point in network can initiate RELEASE
    - » out-of-band, single cell, on "signaling VC". Processed hop-by-hop
- RELEASE contains essential information:
  - Flow ID, QoS class
  - VC id

AT&T

14

Multicast

- ◆ Multipoint-to-multipoint an integral part of UNITE signaling
- ◆ Allow for both “root/core initiated” and “leaf-initiated” joins
- ◆ QoS setup mechanism potentially allows for different QoS on different branches of multicast tree
- ◆ Both leaf-initiated and core-initiated QoS requests “slalom” QoS requests through switches (option 1)
  - – find minimum QoS for the path from “sender” or “core” to receiver

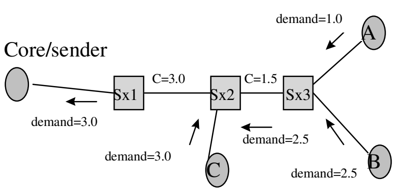


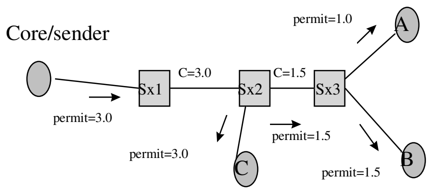
- ◆ Source/core init. joins issued for each destination added.
- ◆ “marker” as part of 3-way handshake regenerated at branch point
- ◆ switches maintain Flow id as multicast call reference

AT&T

15

Multicast QoS Flows





- ◆ Leaf initiated QoS requests carry demand upstream
- ◆ Branch points carry: max ($demand_i$) upstream
  - – different leaves may request at different times
- ◆ Permission returned: min (offered_capacity, $demand_i$) at that branch
  - – offered capacity: could be sender requirement for flow or link capacity
- ◆ Simple extension of QoS processing for unicast flows
  - – each switch maintains state: “demand” and “permit” for each branch

AT&T

16

Summary

- UNITE separates Basic Connectivity from QoS management
- Achieves efficiency while retaining considerable flexibility
  - individual flows may be customized independently of others
  - control overhead is associated with that flow only
- Separating pieces that need efficiency from that requiring flexibility
  - fits in with the desired separation of switching from controller functions
  - easy incorporation of connection-specific handlers and features
- Accomodation of different types of QoS signalling architectures
  - ATM's source initiated
  - RSVP's receiver initiated
- Fits with OpenSig's approach

AT&T