Channel Islands - joining the dots between content distribution & multicast

Jon Crowcroft

Jon.Crowcroft@cl.cam.ac.uk

`http://www.cl.cam.ac.uk/homes/jac22`

May 21, 2002

Networks 2002

Channel Islands

This is a two part talk. Part I is about overlay networks, and how networks evolve in general, through the well-worn example of multicast networking and transport. Part II is about applications and middleware and how they might use multicast as it evolves.

More specifically, and as an *example* of how incremental deployment *may* be done, this talk is about group communication services at the lower layers, and the tension between the pressing need for scalable services and the back-pressure from network providers for stability lead to a form of network deployment known to modern Darwinist theory as *punctuated equilibrium*

We examine then how to make the sudden changeover this type of scenario entails, smoother, through the use of some of ideas that might be exported to higher levels such as overlay networks and reflective open interfaces):

Abstract for part I

The Reliable Multicast Transport group in the IETF http://www.ietf.org/html.charters/rmt-charter.html has taken a radical approach to designing a set of multicast protocols to suit the variety of application and service requirements for group communications.

Instead of designing monolithic protocols (in the style of TCP) a family of protocol profiles is being evolved, together with a set of building blocks. The building blocks are somewhat like micro-protocols , with very open interfaces. There are three one-to-many protocols that are being specified using these, namely a NACK-based protocol, a Tree-based ACK protocol and an Open Loop protocol that uses Forward Error Correction. Building blocks include Generic Router Assist, a tree-building protocol, a layered coding transport scheme, and two novel congestion control schemes.

We note that the set of solutions entail a spectrum of mix of router changes and end system functionality. Interesting results (e.g. Papadopoulos) show that the majority of the performance gains are achieved by edge deployment of router changes only! This bodes well for the future of incremental network service deployment.

Abstract for part II

These are suitable for content distribution, streaming media, software or events, as well as being composed together to form many-to-many services later.

Important interactions with other layers include considerations of: Group Security (access control, authentication, key distribution and re-keying for privacy services etc etc). Internet multicast routing (e.g. dense, sparse or source specific, or lack of multicast and inter-domain problems of MSDP, and general state and router cost of IGMP).

Part I:- Network and Transport

History

IP multicast was devised by Steve Deering at Stanford/PARC, in 1988.

It delivers any-to-many across a variety of distribution tree types, built by an enhanced IP routing layer. Source, Shared and hierarchical trees can be built, relatively efficiently. The algorithms are quite general (e.g. CMU work on application layer multicast used same as DVMRP- SCRIBE used OCBT, effectively:-)

It's best effort. So you need a multicast equivalent of TCP.

What is *equivalent* is the big question.

The IRTF Reliable Multicast Research Group spend a while chasing the holy grail, and then decided "One Size Does Not Fit All":

General Purpose Reliable Multicast is an oxy-moron.

Three particular protocols emerged from the sandstorm... ... ... PGM, RMTP and ALC, which we will discuss more later.

Barriers to Deployment

Christophe Diot et al at Sprint wrote eloquently about the barriers to deployment:

- Economic models
- Traffic Engineering
- Security
- Application Pull
- Diagnostic Tools
- Provider Skills
- I would add: network implementation maturity

However, all this applies to IP unicast too!

IP was once an overlay on the POTS/PSTN.

Overlays are a generic approach to evolving systems.

Successful overlays migrate into the infrastructure.

Too early a migration is a bad idea (viz multicast, mobility, QoS)

So, thats the real story about multicast.

Now, what can we do (assuming we ignore this minor deployment problem) to make it ready for eventual prime time???

Reliable Multicast

So we've done a lot to make IP multicast routing work well. Apart from RTP based applications, what other class of applications do we see in the Net?

Well, clearly reliable ones dominate (HTTP/TCP dominates unicast for now).

What can we do to make life painless for the application writer (make multicast as easy or easier to use than unicast?)?

Design Space

When we looked at `RMTP`, `PGM` and `RLC`, we found some ideas that could be useful in other, future protocols. This suggested that we should disassemble the 3 protocols, and reverse engineer them out of pieces, but leave the pieces available to other people.

Having worked on micro-protocols (e.g. HIPPARCH project, the *x-kernel*, *click-router*, etc), and also having used this approach for realtime multicast and unicast multimedia (`rtp`, `rtcp`, `sip`, `sap`, etc), we decided to take it to the limit.

However, we *also* decided that any-to-many applications were not compelling in the *Big I* Internet. We decided to limit our scope to one-to-many, bulk transfer (i.e. not "RPC for groups", or p2p (e.g. game) transport protocols).

Having said that, it is not obvious that the pieces aren't also useful for many to many, with a higher level coordination.

Non Requirements

What it does mean is that we don't have to solve the atomic broadcast, or global ordering problem (*pace* isis/horus), but can concentrate on partial ordering in a one-to-many stream.

We also decided that authentication, access control and privacy could *probably* be added along side, although we keep a watching brief on this. It must be said that denial of service and traffic analysis are more of a concern with the IP-multicast underlay model that with unicast.

Since we started, PIM-SSM has emerged at the IP layer which, with IGMPv3 (source specific join, if authenticated) goes a long way to making the IP multicast deployment less dangerous for service providers, but also tends to limit us to one-to-many anyhow, and with no address/service advertisement, implies long lived flows.

Requirements/Design Space

RFC 2887 defines these:

- Does the application need to know that everyone received the data?
- Does the application need to constrain differences between receivers?
- Does the application need to scale to large numbers of receivers?
- Does the application need to be totally reliable?
- Does the application need ordered data?
- Does the application need to provide low-delay delivery?
- Does the application need to provide time-bounded delivery?
- Does the application need many interacting senders?
- Is the application data flow intermittent?
- Does the application need to work in the public Internet?
- Does the application need to work without a return path (e.g. satellite)?
- Does the application need to provide secure delivery?

Non functional requirements:-)

So then we have requirements that arise indirectly from the nature of the IP multicast model:

- ack implosion leads to nack aggregation
- tree dependence
- router support (# groups, packet filter, leave/join rate, special?)
- rate and congestion management - e.g. single or multiple rate

Building Block Rationale

Being computer scientists bought up in the 80s, we were 2 decades ahead of the comms hackers, but 1-2 decades behind modern software engineering. But we do believe in a few late 20th century principles:

- Specification Reuse.
- Reduced Complexity.
- Reduced Verification and Debugging Time.
- Easier Future Upgrades.
- Common Diagnostics.
- Reduces Effort for New Protocols.
- Parallelism of Development.

However, we did recognize risks too:

- Delaying Development
- Increased Complexity.
- Reduced Performance.
- Abandoning Prior Work

Functional Decomposition

```
Data Reliability (ensuring good throughput)    |
  | - Loss Detection/Notification
  | - Loss Recovery
  | - Loss Protection

Congestion Control     |
  | - Congestion Feedback
  | - Rate Regulation
  | - Receiver Controls

Security

Group membership       |
  | - Membership Notification
  | - Membership Management

Session Management     |
  | - Group Membership Tracking
  | - Session Advertisement
  | - Session Start/Stop
  | - Session Configuration/Monitoring
```

Building Blocks

- nack based reliability
- FEC coding
- congestion control
- generic router support
- tree configuration
- data security
- common headers

Protocol Instantiations

PGM (uses GRA) - novel features - e.g. active nets (dynamic filters), and *window advance with time* (was original Reuters/TIBCO/Cisco TIB requirement, after all:-)

RMTP2 (uses TRACK and NORM and can use GRA) - novel features -builds a parameterized tree amongst peers.

ALC (uses FEC and can use GRA) - novel features - multi-rate, massive scale, no return path needed etc etc

Summary & Conclusions to Part I

Did it work?

We don't know yet - we are *just* doing the instantiations.

It worked in the sense of making our documents (see `http://www.ietf.org/html.charters/rmt-charter.html`) clearer.

We haven't seen anyone try to take a BB and use it in another layer - please do ... ...

Note on overlays as incremental deployment for other services: It is not at all clear whether hard performance guarantees (int-serv) could be offered as an overlay service (overlay QoS). However, it is quite possible that statistical guarantees could be achieved through overlay routing, admission, policing and shaping. Watch this space!

Future Stuff Part I...

- Security - biggest problem is re-keying.. .. ..
  1. Access control and authentication of sender (and receiver in IGMP join or invite) is easy
  2. Privacy looks easy until you realise that to scale, you use PKC for join,m then a shared key for data distribution, but re-keying when someone *leaves* scales badly, even with hierarchy of 1 way functions and key graph... DOS seems hard (esp. for receiver driven protocols without auth/gra)

- APIs will be interesting - e.g. reflection... ... ...see next part of talk

Big Picture Stuff Part I...

Network evolution is a rare thing (analog to digital PSTN tool 30 years...still happening)

More common to work though overlays and parallel development.

Multicast needs to be deployed in this way - the Mbone was the right path, and we forced the change to native mode too soon.

Think about Vinyl to CD, VHS to DVD, Book to Phone, Film to TV, Library to Web and so on

Stay virtual as late as possible!

Part II:- Applications and Middleware

One person's middleware is another persons application...

News/Bboard/Mail lists

NNTP is a natural for multicast - in fact this has been noted (c.f. Usenix paper on drinking from the firehose).

Unit of transfer is a posting

Ordering not needed outside this

Not really bulk (total NNTP traffic is ¡¡ 1% of the net)

Not hard ,but also not very interesting...

Software Distribution

Jim Gemmell noted that Microsoft corp. saturates a 1.2Gbps link 24*7*52.

But if you total the capacity on www.Microsoft.com, and take 1 year to send it, you get 10kbps.

Mirror sites deal with time shifting.

Looks like a case for ALC, but PGM or RMTP also work very well - depends on customer sites (satellite, varying rate, behind ALGs/Firewalls/Proxies, Tivo, Settopbox etc etc)

Share Trading

This is what PGM was designed for - single rate.

Sell by date dealt with by window.

Buy by date. - is a different session, so can be dealt with otherwise completely

Note PGM (as an exceptional behaviour) does have a notion of local repair servers, but poorly developed compared with say RMTPII.

Games

Right now there are about 3500 half-life and quake servers out there with paying customers. They aren't multicast. There could be a LOT more game players if they were peer2peer, but:

Games, or DIS are just about the hardest case.

you need at least *ABCAST* semantics

you need minimum rate

you need delay bounding

General cheating problem (c.f. nice work by Levine et al)

Won't happen on the Internet as we know it, Jim, well not yet anyhow...

Event Distribution

c.f. scribe..... could use lower level multicast if reliable and timely

looks like a case for PGM...with local repair

probably not really ALC (although multi-rate events might be ok if the rate was semantically associated with sub-streams...actually TIBNET has a hack for this in PGM too)

almost certainly (for 2.5 and 3G phone use) are behind ALG/proxy, so RMTP works too...

Generic Peer-to-peer

Not clear - do s/w, music/dvd rip/pirates overlap in time?

eternity and oceanstore might use ALC though...

see also CAN for multicast from ratnasamy at UC Berkeley.

Aside on Content Name Spaces

It is clear that apart from multicast, one thing that all of these applications have in common is the requirement for a content naming and subscription specification service.

We need to be able to advertise, and withdraw graph-based names - these specify the relationship of content on a multicast channel, and associate it with attributes so that subscribers to the channel can express predicates about their *interests* and about performance related parameters such as *proximity*.

For example:

Net News We say we want to read article 27, on bboard alt.foo.

Events We are interested in restaurants near Pisa with tables free for 4 at 8pm.

Software distribution/subscription We would like the RPM for magicpoint 4 for Linux Red Hat 2.4.7, from the highest bandwidth server near Cambridge.

Games We want to play *Half Life* on a server near here with at least 15 other players at Kill Level 99.

A property of such a naming and filtering service is that it should be implemented using the channel architecture itself.

Channel Islands

What if there is only *partial* IP multicast deployment?

Solution space:

- ALGs - reflectors, tree builders
- Tunnels
- Multicast Address Translation (including m2u and u2m)

This latter is a hack of mine - we have NATs so why not have MATs (translate mcast addresses between *realms*. A realm can loan multicast space to someone else that doesn't have it, or hasn't enough addresses, or wants to do privacy, or aggregation with it....

What if we use *deep* reflection?

I.e. pull out behaviours of net, transport and tree building and use them in higher levels (in reflective middleware).

For location, route, loss/repair, ordering, traffic management, etc etc.

Reflective Routing Services

So the idea is that we *create*, *join* groups from receiver locations, as if they use the underlying addresses, and we ask the IP routers, which *know* the underlying topology (link metrics, traffic conditions etc), to *compute* (they know how to do Dijkstra:-) routes to our receiver sets from our sender sets, but, but but

We ask them to allow for non multicast capable regions (i.e. to say where they need someone to replicate packets *for them*.

Need some nice s/w engineering here...

Then we're done.

Summary and Conclusions Part II

We don’t want to re-invent wheels, we want to rediscover which type of wheel is useful where.

The *end to end [to end]** design principle can be re-stated for multicast as “don’t replicate functions that need to know lower level details in higher levels, but don’t expect pervasive deployment of higher level functions everywhere - use open APIs and reflection, and mix and match.”