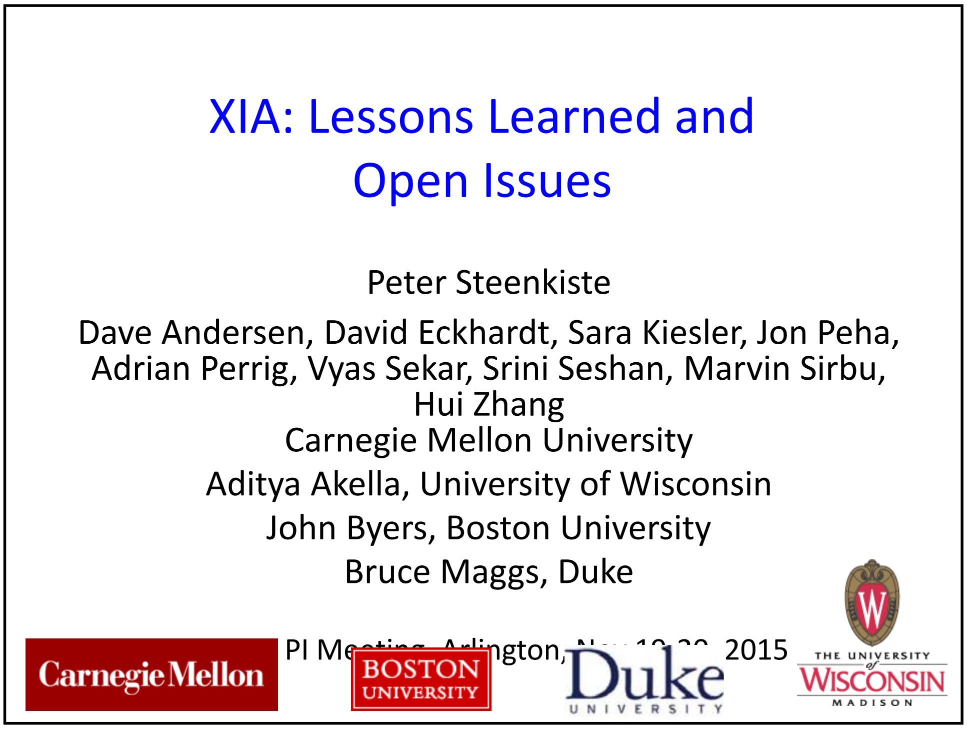


XIA: Lessons Learned and Open Issues

Peter Steenkiste

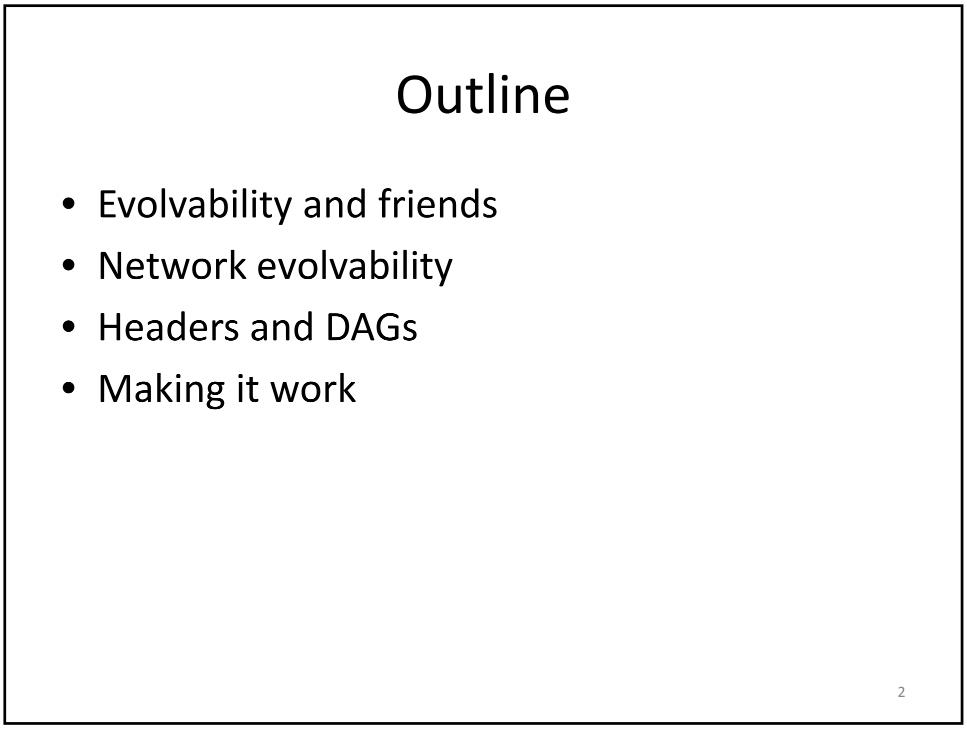
Dave Andersen, David Eckhardt, Sara Kiesler, Jon Peha, Adrian Perrig, Vyas Sekar, Srini Seshan, Marvin Sirbu, Hui Zhang
Carnegie Mellon University

Aditya Akella, University of Wisconsin

John Byers, Boston University

Bruce Maggs, Duke

PI Meeting, Arlington, Nov 19-20, 2015



Outline

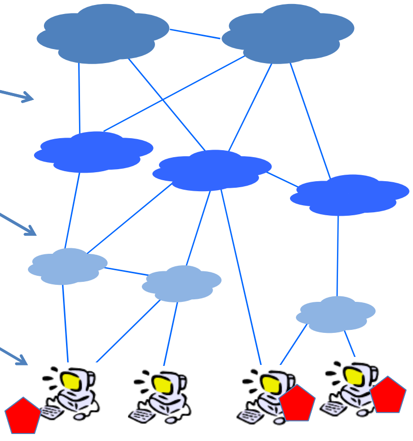
- Evolvability and friends
- Network evolvability
- Headers and DAGs
- Making it work

Evolvability Lessons

- Mechanisms are effective in supporting evolvability
  - Starting with NID and HIDs, added CIDs (content), Scion IDs (path based forwarding), 4IDs (legacy IP), anycast SIDs; working on RIDs, CID variants
- But evolvability mechanisms support other important usage models
  - Customization of networks
  - Discovery function
  - Selective and opportunistic deployment

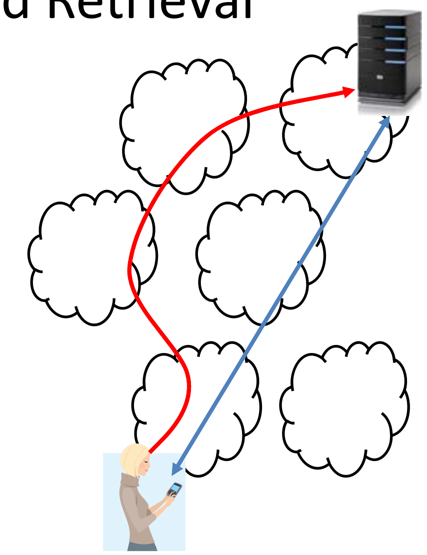
XIA Supports Network Diversity

- XIA in public Internet:
  - Core networks: NID only
  - Eyeball: NIDs plus some value-added services
    - E.g., CID and SID
  - Customer networks can be very diverse
    - Possibly many XIDs
- Edge networks can introduce custom XIDs
  - Unique forwarding or security requirements



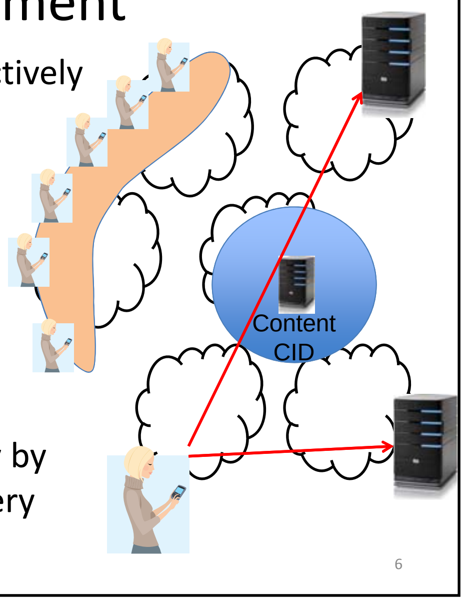
“Fetch” XIDs Support Discovery and Retrieval

- XID first discovers desired destination
  - Anycast, CID, RID
  - Fetch as opposed to send
- Communication session is point to point
  - We use unicast communication (NID/HID)
- Change of end-point addresses is signed
  - Uses intrinsic security



Selective and Opportunistic Deployment

- XIDs can be deployed selectively
  - Business incentives
  - E.g., caching near edge
- Entries can be advertised selective
  - Performance driven
  - Business agreements
  - Custom routing protocols!
- Fallbacks eliminate penalty by allowing in-network recovery
  - Ok to fail



Evolvability Discussion and Issues

- XIDs can represent wide range of “things”
  - Forwarding semantics to “destination”; path-based forwarding; embedding of arch. concepts
- In-network error recovery using fallbacks is very powerful
  - Failures, opportunism, …
- Intrinsic security has been relatively basic
  - Based on pubic keys, hashes; maybe signatures?
  - Original definition probably too strict

Outline

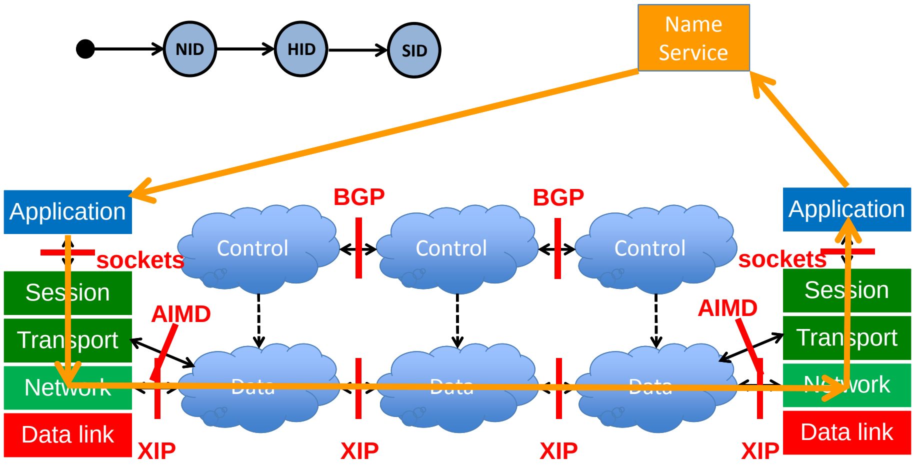
- Evolvability and friends
- Network evolvability
- Headers and DAGs
- Making it work

Goal is Network Evolvability - Network layer not sufficient

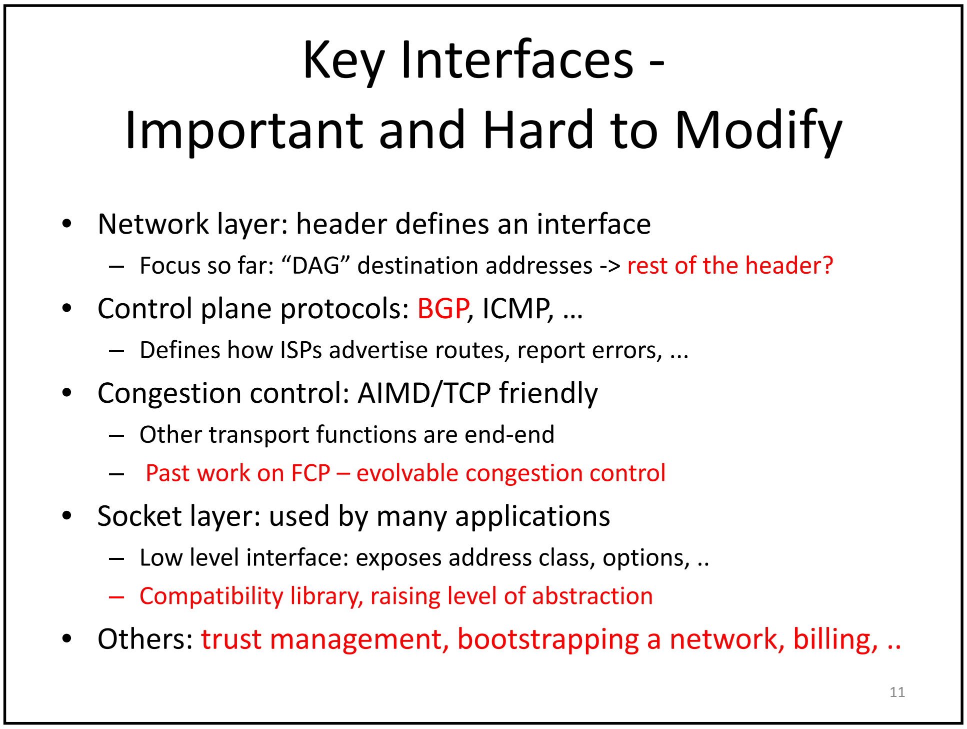
- Observation: BGP has not changed much
  - Last revision is early 90's
  - Proposed extensions are hard to deploy
- Lots of progress in congestion control
  - But has to be TCP-friendly, defined in ad hoc way
- Socket API has changed … by adding new calls
  - Old calls never go away, e.g., ftp
  - Address type is exposed to application

9

What Interfaces Do We Care About?

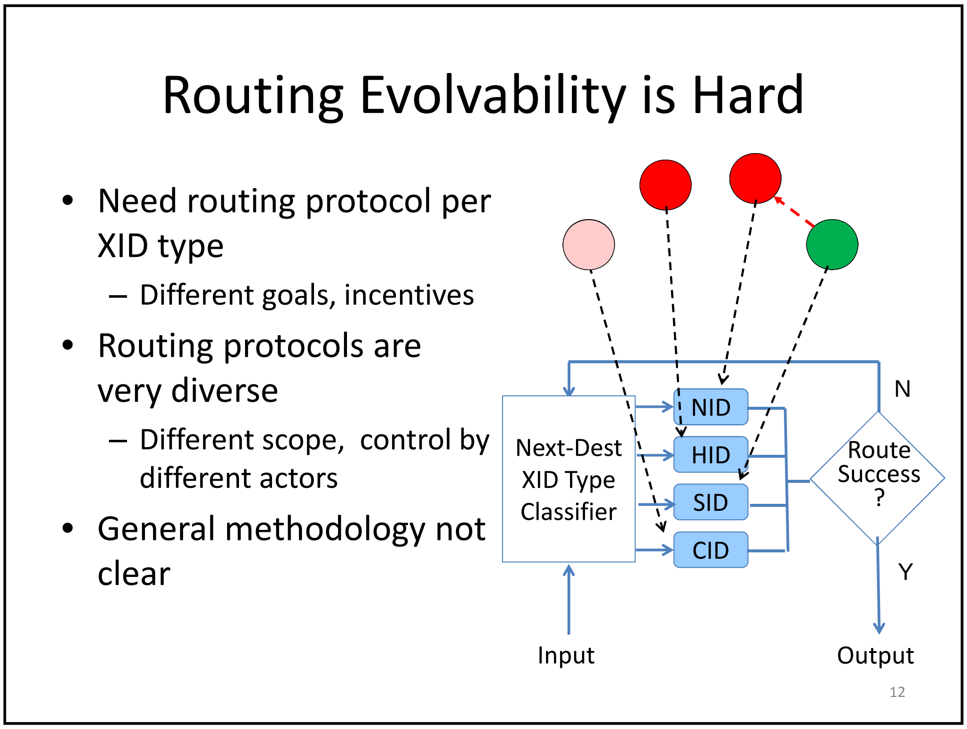


10



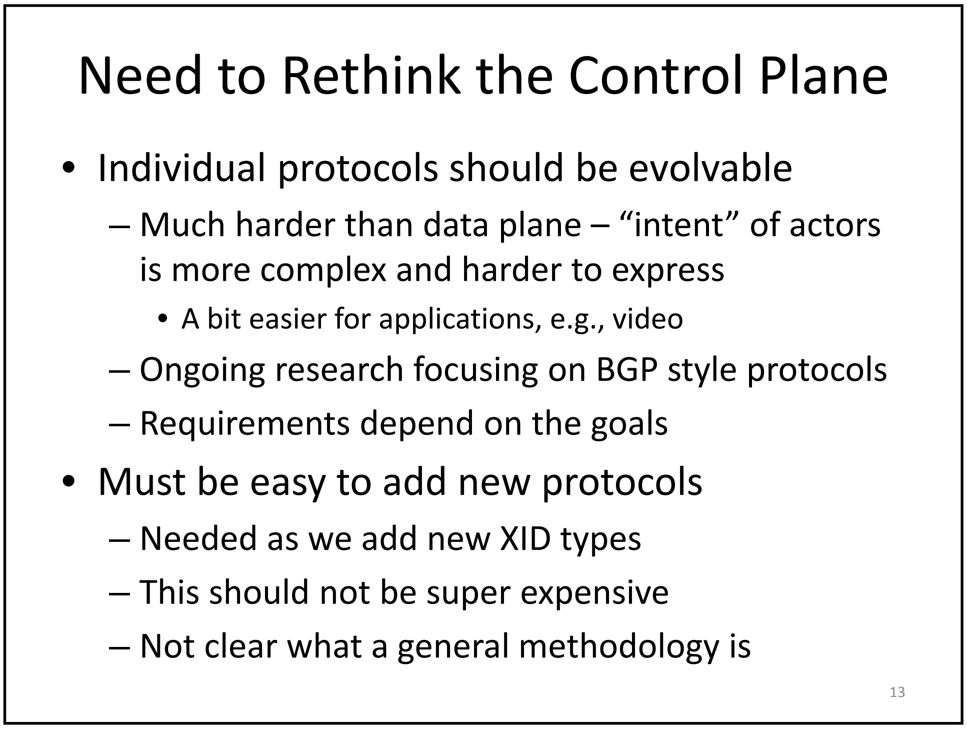
Key Interfaces - Important and Hard to Modify

- Network layer: header defines an interface
  - Focus so far: “DAG” destination addresses -> rest of the header?
- Control plane protocols: BGP, ICMP, …
  - Defines how ISPs advertise routes, report errors, …
- Congestion control: AIMD/TCP friendly
  - Other transport functions are end-end
  - Past work on FCP – evolvable congestion control
- Socket layer: used by many applications
  - Low level interface: exposes address class, options, ..
  - Compatibility library, raising level of abstraction
- Others: trust management, bootstrapping a network, billing, ..



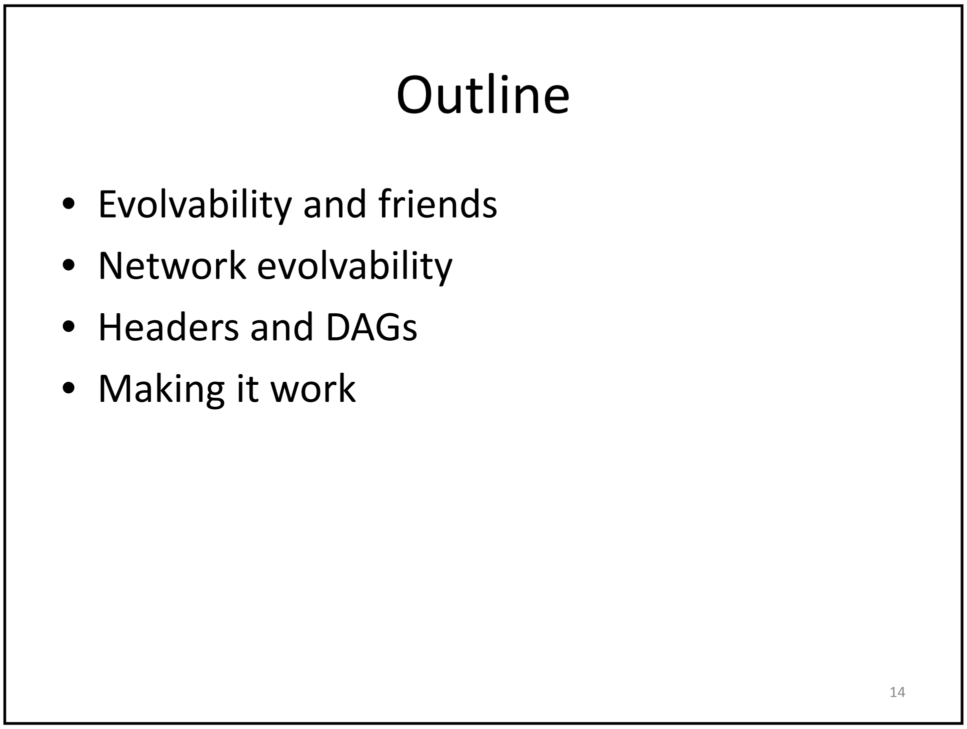
Routing Evolvability is Hard

- Need routing protocol per XID type
  - Different goals, incentives
- Routing protocols are very diverse
  - Different scope, control by different actors
- General methodology not clear



Need to Rethink the Control Plane

- Individual protocols should be evolvable
  - Much harder than data plane – “intent” of actors is more complex and harder to express
    - A bit easier for applications, e.g., video
  - Ongoing research focusing on BGP style protocols
  - Requirements depend on the goals
- Must be easy to add new protocols
  - Needed as we add new XID types
  - This should not be super expensive
  - Not clear what a general methodology is



Outline

- Evolvability and friends
- Network evolvability
- Headers and DAGs
- Making it work

Headers – How Hard Can it Be?

- XIA header modeled after IP header
  - Source and destination address (DAGs include both address and SID of the socket)
  - Version, TTL, length
- Source addresses used for accountability
  - Role can be separate from return address
  - A bit tricky fallbacks – need more experience
- Need indication whether content is cacheable – should be separate from source address

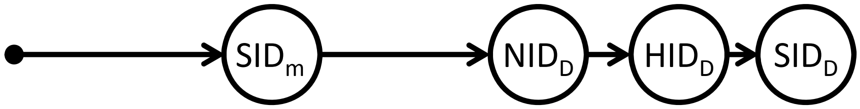
15

DAGs – Manageable but a Bit Messy

- Constructing good DAGs is a bit tricky
  - Should not be left to users or applications
- General idea: destinations (services) advertise their DAG but senders can modify it
  - E.g., insert in-path services, policy issues, ..
  - Should be done by a package
- Client node get DAG from network
  - Customization done using place holder XIDs
- Policy compliance checking easy for simple DAGs
  - Limited experience with complex DAGs

16

What Does This mean?

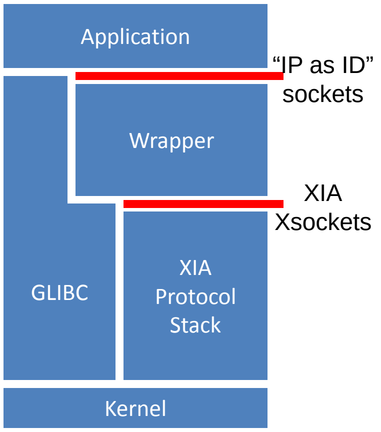


- SID_m is visible (non-transparent) middlebox
  - Trusted by source since it inserted the SID in DAG
- Can be used for any traditional middlebox
- Also used for custom forwarding or address translation services
  - Rendez-vous mobility service, inserting Scion ID
  - More general: delegating address creation

Outline

- Evolvability and friends
- Network evolvability
- Headers and DAGs
- Making it work

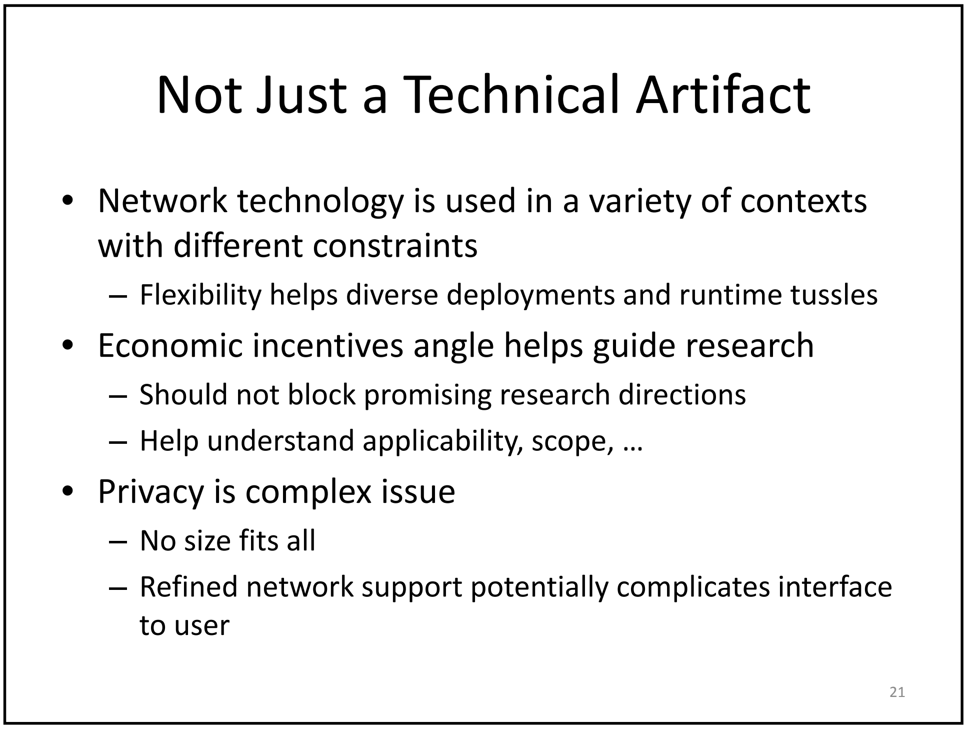
Porting Applications is a Pain



- And it is mostly not research!
- XIA xsocket = ~IP sockets + get and put chunk
- Solution: translates IP socket calls automatically and add CID calls manually
  – The latter is research!
- Solution is based on a wrapper library
  – (IP, socket) -> XIA DAG
  – Kind of like a weird NAT
- Discussion? Experiences

Protocol Stack Rethinking

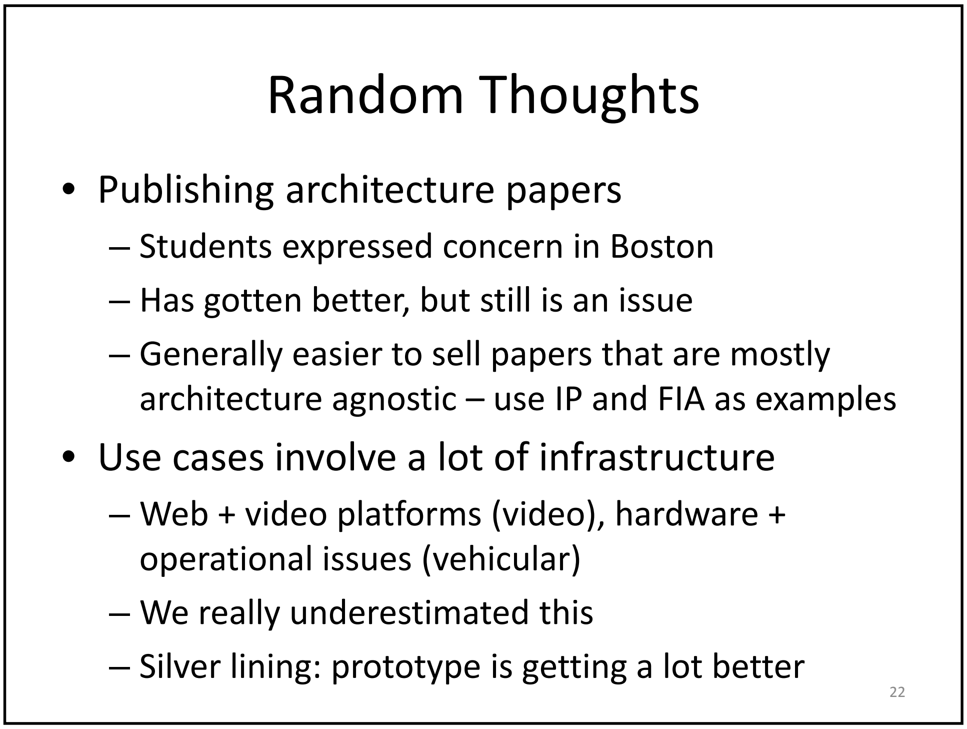
- Some XIDs push application layer concepts to the network layer
  – Content, services, "discovery", …
  – Jumps over the transport layer
- These concepts are application independent
  – But in practice applications need application specific meta data, e.g., HTTP vs GetChunk CID
- General question: difference between an XID and a protocol – how much header needed?



Not Just a Technical Artifact

- Network technology is used in a variety of contexts with different constraints
  – Flexibility helps diverse deployments and runtime tussles
- Economic incentives angle helps guide research
  – Should not block promising research directions
  – Help understand applicability, scope, …
- Privacy is complex issue
  – No size fits all
  – Refined network support potentially complicates interface to user

21



Random Thoughts

- Publishing architecture papers
  – Students expressed concern in Boston
  – Has gotten better, but still is an issue
  – Generally easier to sell papers that are mostly architecture agnostic – use IP and FIA as examples
- Use cases involve a lot of infrastructure
  – Web + video platforms (video), hardware + operational issues (vehicular)
  – We really underestimated this
  – Silver lining: prototype is getting a lot better

22

Questions?
Discussion?

More Topics (short)

- Different XID types need different ctl protocols – reflects semantics – not just BGP++
  - Several of these protocols look, well, quite different – any general concepts?
- Services allow visible in-path services, specialized forwarding semantics (based on header, …)
  - Changes architecture