Finally, a Use for Componentized Transport Protocols

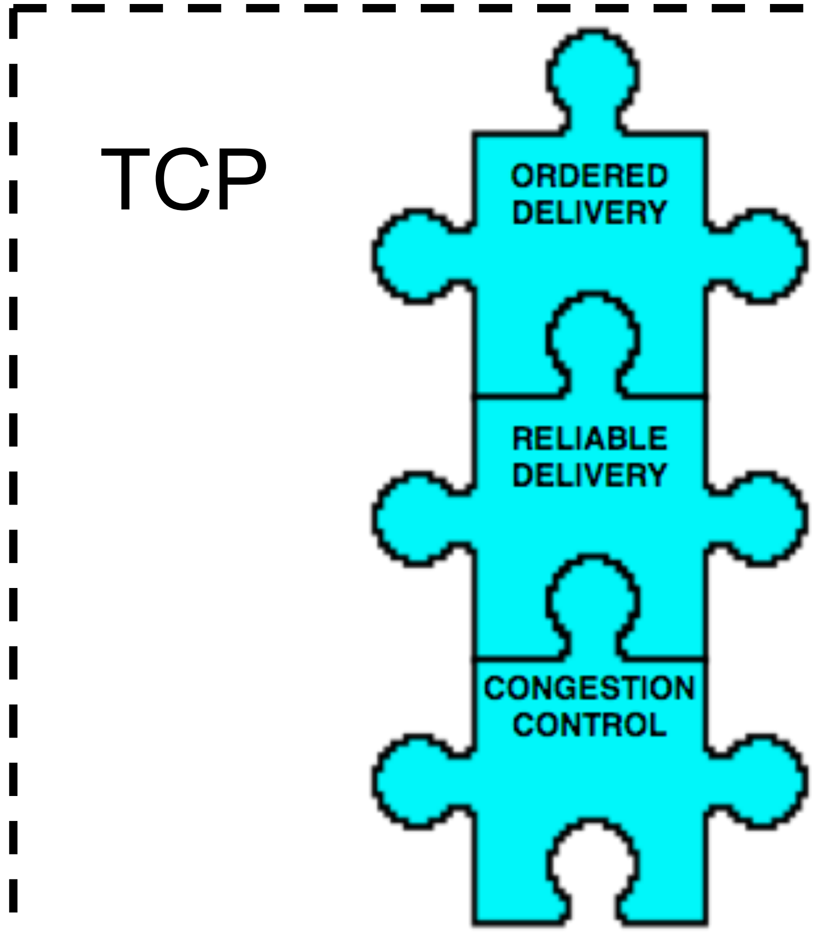
Tyson Condie, Joseph M. Hellerstein,
Petros Maniatis, Sean Rhea, Timothy Roscoe
U.C. Berkeley and Intel Research Berkeley

Roadmap

- History
- What's different now?
- New uses for componentized transport protocols
- Conclusion
- Ongoing work

Componentized Protocols?

Decomposing transport protocols into a set of reusable building blocks that can be recomposed in different ways depending on application and network properties



A Brief History of Componentized Protocols

- x-Kernel system
  - Export user-level protocol objects to compose into more general communication services
- Morpheus programming language
  - Object oriented programming support for protocol objects
  - Compiler time optimizations over generated protocols
- Prolac Protocol Language
  - Expression language for developing complete protocols

[illegible]

- Componentized protocols never caught on
  - Most applications were satisfied with point-to-point protocols
    - TCP, RTP, SCTP, DCCP, etc.

Why Now?

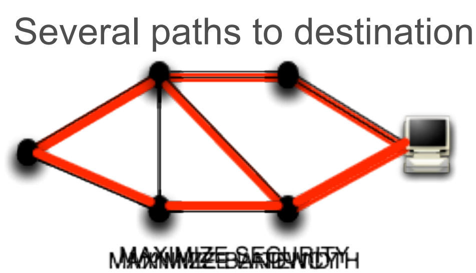
- Increasing popularity of overlay networks
  - DHT, BitTorrent, Akamai, Narada, Freenet
- Overlay networks have a broad design space
  - Nodes play the role of client, server, and router
- Most protocols today are tuned for point-to-point communications
  - Overlay requirements go beyond point-to-point model
  - Forces overlay programmers to develop their own handcrafted transport layer
- We have built a componentized protocol framework using a dataflow abstraction
  - Does it meet the set of opportunities and requirements of overlays?
  - Does it provide a programmer friendly framework?

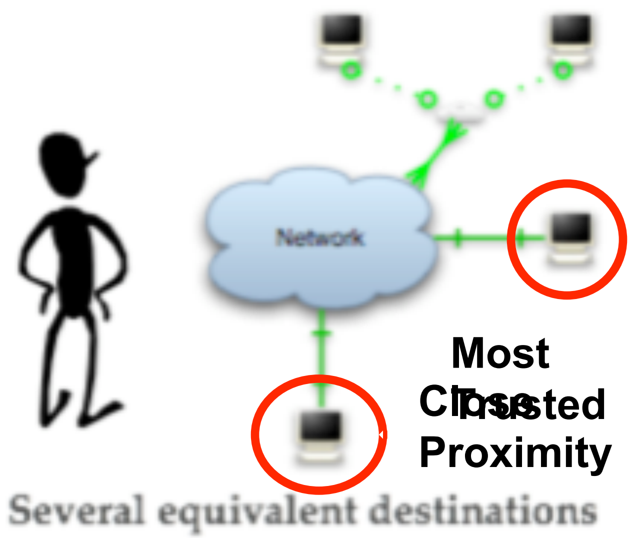
Every good work of software starts by scratching a developer's personal itch

The Cathedral and the Bazaar, Raymond

Application-level Routing Freedom

- A message need not be bound to a single destination or path
  - Several equivalent destinations
  - Several paths to get to a destination
- Fine grain application control over where a message is sent





Alternative Congestion Control

- **Per hop** congestion control
  - Recursive routing
    - UdpCC connection for each neighbor node

S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. **Handling Churn in a DHT.** In *Proc. of the 2004 USENIX Technical Conference*, Boston, MA, USA, 2004.

- **Aggregate** congestion control
  - Short lived connections
  - Connections that send very little traffic
  - Iterative routing (a la MIT Chord)
    - Next hop discovered during the lookup

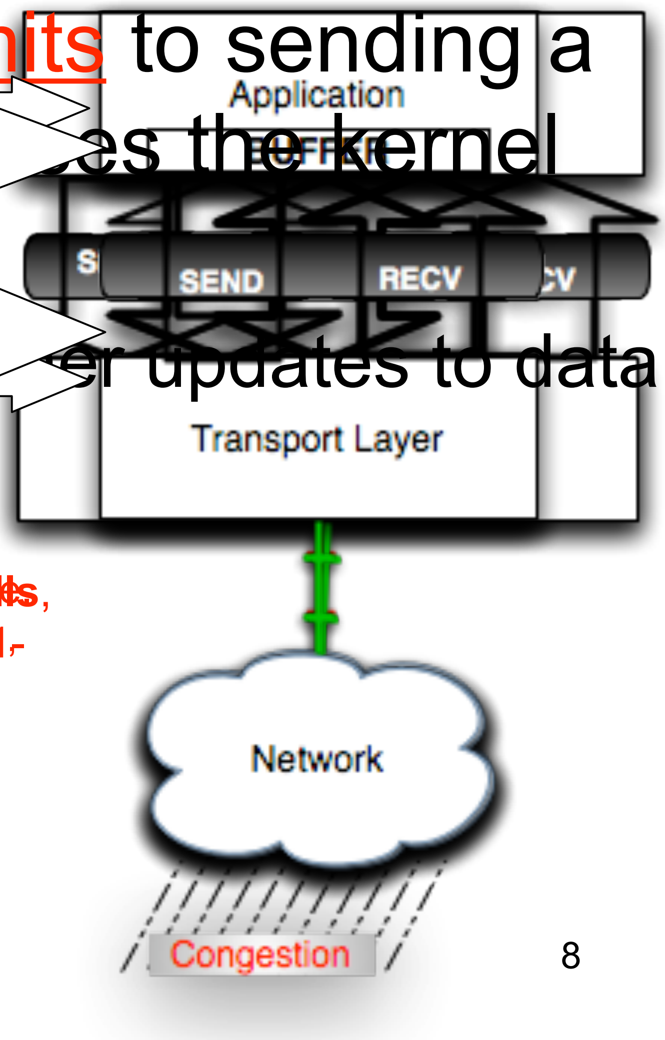
F. Dabek, J. Li, E. Sit, F. Kaashoek, R. Morris, and C. Blake. **Designing a DHT for low latency and high throughput**. In *Proc. NSDI*, 2004.

Late Data Choice

- In TCP an application commits to sending a packet when it ... the kernel boundary
  - Kernel boundary ... later updates to data

Move Buffer Management to Application

Invert call semantics and use upcalls to signal ready



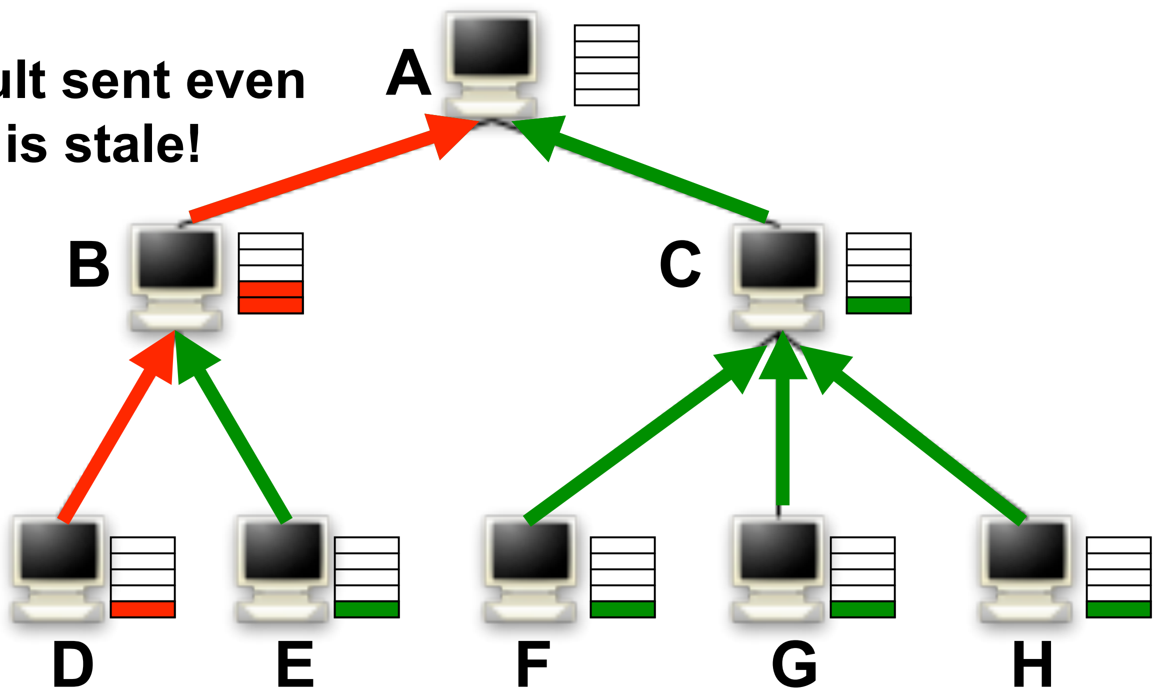
David D. Clark, **The structuring of systems using upcalls**, ACM SIGOPS Operating Systems Review, v.19 n.5, p.171-180, Dec. 1-4, 1985

J. Mogul, L. Brakmo, D. E. Lowell, D. Subhraveti, and J. Moore, **Unveiling the transport**, *SIGCOMM Comput. Commun. Rev.* 34(1):99-106, 2004

Late Data Choice

- Application sensitive to stale results
- Transmission costs are high
  - Late data choice ensures the most up-to-date computation is sent

Prior result sent even though it is stale!



Roadmap

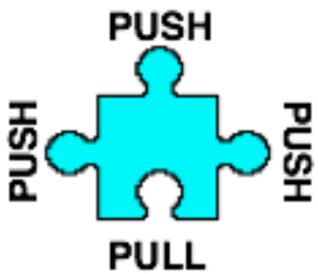
- History
- What's different now?
- New uses for componentized transport protocols
- Conclusion
- Ongoing work

Benefits to Componentized Protocols

- Fine grain protocol modifications
  - Alternative congestion control
  - Message/packet level reliable delivery
  - Custom packet scheduling algorithms
- Transport layer more knowledge visible
  - Late data choice
  - Transport state can aid failure detection, replica selection, load balancing decisions, etc.
- Encode domain knowledge in the transport layer
  - Overlay routing logic
  - Message semantics

Componentized Protocols using a Dataflow Abstraction

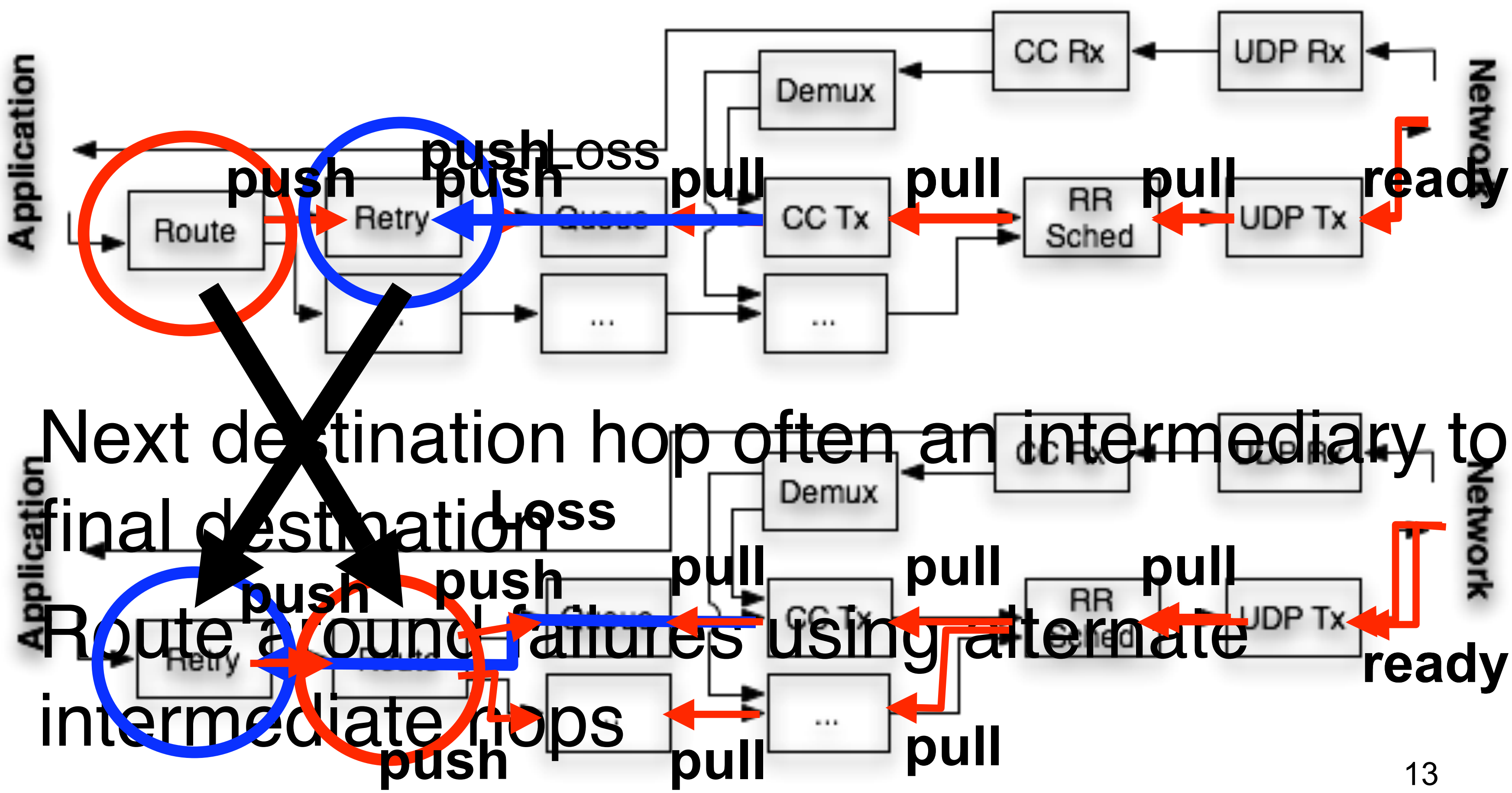
- Graph model places elements at the vertices
  - Elements abstract code into modular units that perform a specific task
  - Elements export push or pull **interface**



- Graph structure orders data transformations
  - Traditional protocols follow stack ordering
  - Dataflow more general
    - Protocol semantics encoded into the graph structure

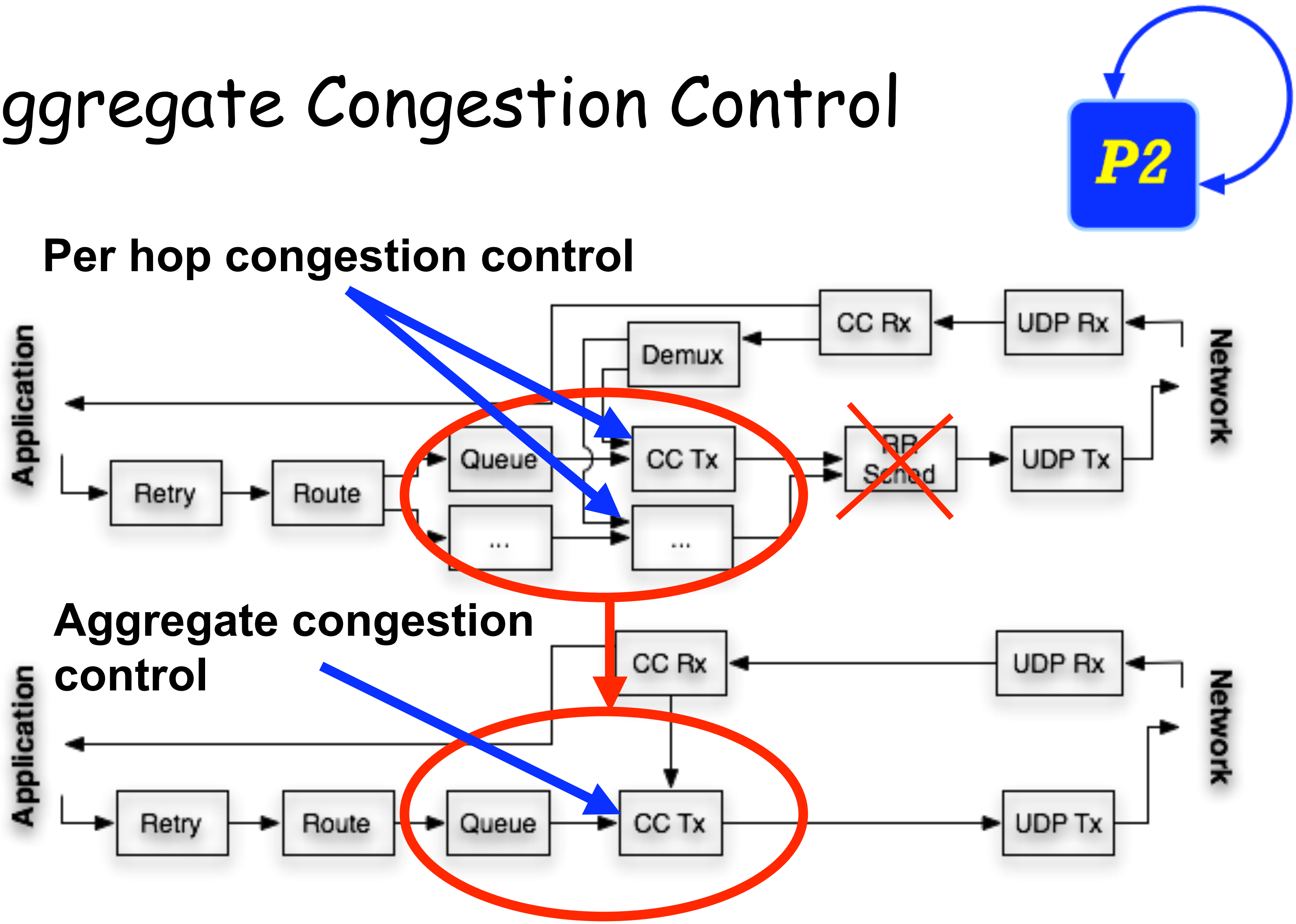
E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. **The Click modular router.** ACM Trans. Comput. Syst., 18(3):263-297, 2000.

Application-level Routing Freedom



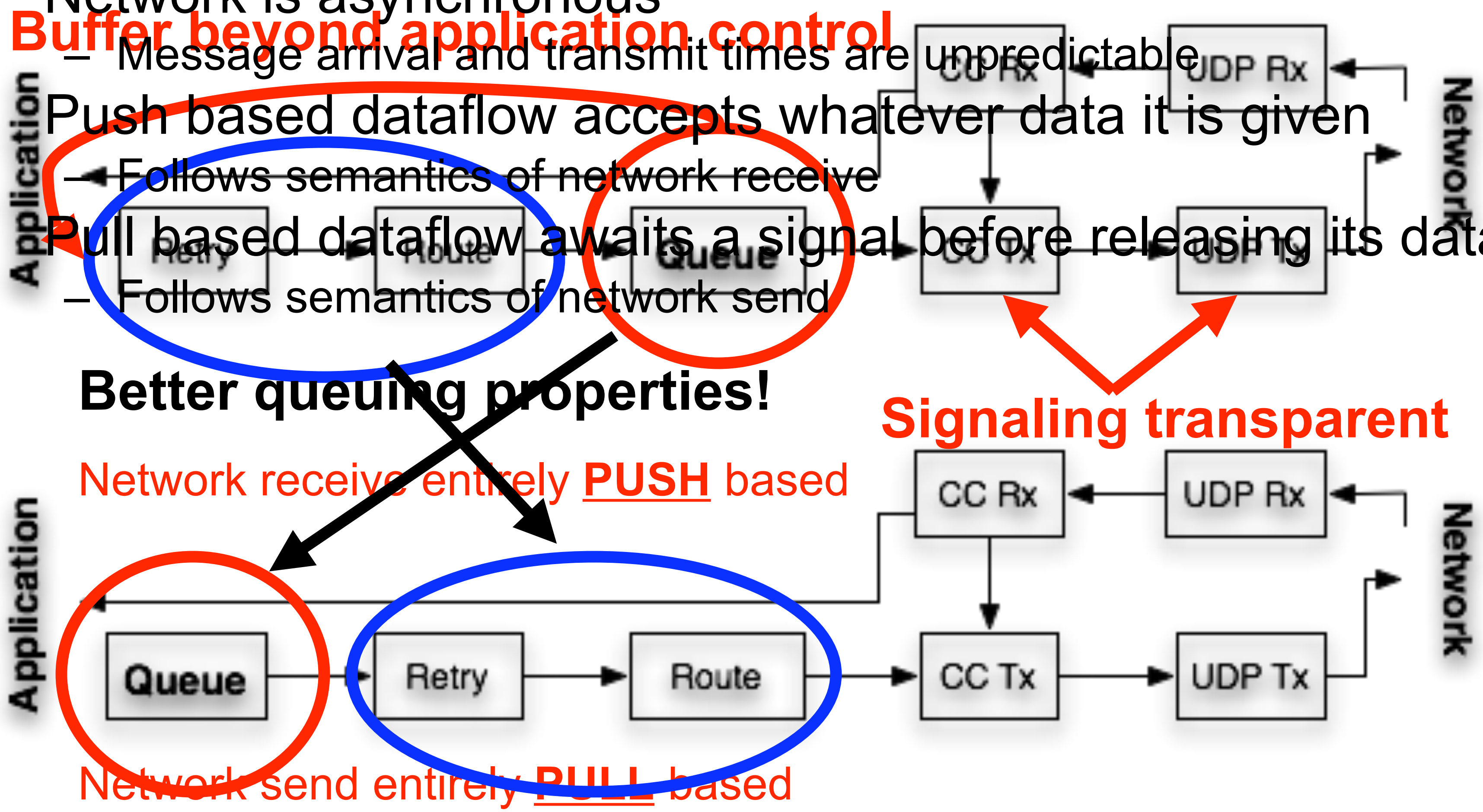
- Next destination hop often an intermediary to final destination
- Route around failures using alternate intermediate hops

Aggregate Congestion Control



Late Data Choice

- Network is asynchronous
  - Message arrival and transmit times are unpredictable
- Push based dataflow accepts whatever data it is given
  - Follows semantics of network receive
- Pull based dataflow awaits a signal before releasing its data
  - Follows semantics of network send



We can do all this in P2

- P2: A query processor for constructing overlays
  - Uses a declarative language for specifying queries that describe overlay properties/invariants
  - Queries compiled into a dataflow graph
- P2 dataflow model extends into network stack
  - Satisfies our transport layer needs for building overlays
  - Blurs boundary between application and transport

B. T. Loo, T. Condie, J. M. Hellerstein, P. Maniatis, T. Roscoe, and I. Stoica. **Implementing declarative overlays.** In *Proc. ACM SOSP 2005*.

Conclusion

- Overlays offer many new design decisions
  - Functionality requirements go beyond the scope of current monolithic transport services
  - Requirements well suited to componentized protocols
- Componentized protocols erase the functionality boundary
  - Can encode the application and transport layer with the right set of features
- Dataflow is an instance of componentized protocols
  - Flexible glue layer between network and application
  - Code reuse through graph modification

Ongoing Work

- Declarative language for transport layer
  – Translate high level invariants into supporting dataflow(s)
- Automatic static generation of dataflow graphs
  – Each semantically equivalent dataflow can offer certain application and network tradeoffs
  – Cost model chooses an optimal dataflow to install
- Runtime reconfiguration / reoptimization
  – What kinds of modifications and how are they triggered?
  – What kinds of statistics would aid in this effort?

Thank You!

For more information please visit our website
http://p2.cs.berkeley.edu/