The Internet: Past, Present, and Future:

TCP/IP & NDN

Alexander Horn

DePaul University SNL/CDM GIS

Mar 22 2011 – First Draft

July 1 2011 - Final Content Revision

September 2011 – Formatting / Editing Revision

November 2011 – Multicast Addition

1 Introduction & Background

The Internet has grown from a military-funded academic experiment in robust communications in 1969 to join water and shelter as a fundamental human right within a generation (La Rue, 2011). Yet the technology also has a dark side, not the least of which is has resulted in the diminishment of privacy, ease of physical infrastructure attack, spam (excess useless traffic), and facilitating copyright evasion.

Can these emergent and growing problems be fixed within existing infrastructure - or are these failings so implicit in the Internet architecture that avoiding them requires a fundamentally different Internet architecture? Let’s explore, starting with the origins of the existing Internet.

1.1 Internet overview

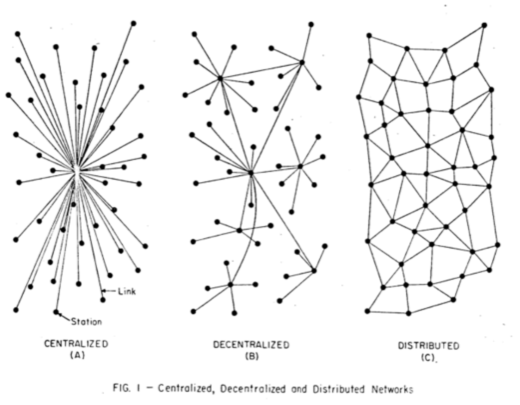
Link
Station
CENTRALIZED (A)
DECENTRALIZED (B)
DISTRIBUTED (C)
FIG. 1 – Centralized, Decentralized and Distributed Networks

Figure 1-3 (Baran 1962)

Before the Internet, there was Arpanet. Arpanet was designed to take advantage of existing telephonic infrastructure in a manner that would allow robust communication even with significant degradation of that infrastructure (Baran 1962). This was done by means of introducing packet switching on a distributed network, as opposed to circuit switching on a decentralized network. Circuit switching had been around since late 1880s and was used for telephony, as it allows two endpoints (telephones) to have their own communications channel on-demand, by simply requesting it (via operator, or rotary, or tone). However, the physical resources for that channel were only of use to the two parties during the connection, and if any break in the circuit between

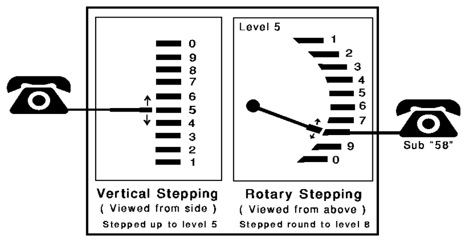

Figure 1-2 (SEG 1996)

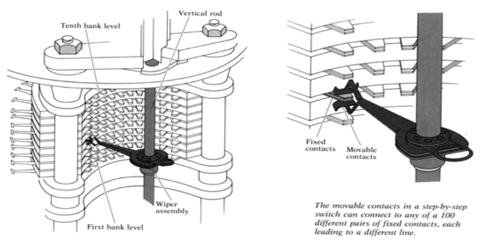
Tenth bank level
Vertical rod
Wiper assembly
First bank level
Fixed contacts
Movable contacts
The movable contacts in a step-by-step switch can connect to any of a 100 different pairs of fixed contacts, each leading to a different line.

Figure 1-1 (Victor 2002)

them would break the connection. Packet switching allowed multiplexing of connections as well as dynamic routing, such that a 'virtual channel' could be created, and its message diverted around any breaks in the circuit.

Arpanet, linking mainframes at UCLA, UC Santa-Barbara, Stanford, and University of Utah (Forouzan 2006), had proven the viability of packet switching to many skeptics (Clark 1988), and the Internet project soon followed. It was originally designed it merely to link Arpanet and a packet radio network, so that the resources of the Arpanet would be accessible to the users of packet radio (Clark 1988). To serve as an inter-network, it had to have elemental features of all networks, and be able to transduce data in and out of them – it's with these principles that TCP was theorized in 1973 and first implemented in 1977, to be split into TCP and IP shortly after (Forouzan, 2006)

We'll discuss more architectural detail shortly, but it's enough for us now to simply note that the Internet was built in order to provide robust dataflow over existing communication infrastructure, that is, phone lines. It did so by creating a new 'atom' of communication – the datagram – and a new way to move it - packet switching. The datagram is a stateless packet, carrying a fragment of data along with its sender & receiver's address so it can navigate the network (Cerf, Kahn 1974).

1.2 NDN overview

"Significant research projects have been funded in the last years focusing on the definition of novel architectures for the future Internet (e.g. US NSF GENI or EU FIA). In this research arena, content-centric proposals, as Parc's CCN, PSIRP (now PURSUIT), or DONA, aim at redesigning the Internet architecture with named data as the central element of the communication paradigm" (Carofiglio et al 2011). Each of these proposals bring fresh thinking into Internet architecture design by re-centering communication principles around current usage, mostly content dissemination and retrieval; radically changing network architecture by moving content storage and delivery into the network layer itself (Carofiglio et al 2011). This paper focuses specifically on Content-Centric Networking (CCN).

CCN is the name given to a research project started at Xerox Parc in 2003. It's become an open source project (CCNx), and is part of a Named Data Networking (NDN) project under the National Science Foundation's Future Internet Architecture (FIA) initiative. The terms CCN, NDN, and CCNx are used interchangeably in this paper. CCN's creators were graduate students on the original Internet project, writing popular networking tools tcpdump and traceroute, and remain active. Indeed, they went so far as to 'fix the internet' in the late 1980s (with header compression that still runs in 90 percent of Internet routers) – as well as co-authoring VOIP & RTP (Parc, 2011). The result is that CCN is conceived with tremendously intimate knowledge of the Internet as it is today, including the design trade-offs that got us here. As a fundamental new architecture, it can, theoretically, learn from and correct the irregularities.

CCN currently overlays above TCP/IP for experimentation and – but is designed to eventually supplant it via integration w/ existing sub-IP protocols. CCN's central aim is to finally decouple the Internet from the legacy of telephony. What this means is the death of the two-party channel and location-dependence of routing, aspects that have become a protocol burden and are increasingly irrelevant to how the Internet is used today (Jacobsen et al, 2009)

Unlike a telephone call or TCP/IP unicast packet, CCN packets can have multiple receivers rather than just one, contain no source or destination address, and are cryptographically signed at the source. Packets can thus be distributed more efficiently not only in space, but also in time, preserving both anonymity and providence in a way that scales well. Popular data becomes faster to get, and less of a burden on the source, not unlike Bit Torrent and other peer-to-peer networks. This new architecture has promise to reduce Denial of Service, copyright evasion, and spam, as well as improve privacy and security and overall network performance.

However, NDN is in a state not dissimilar to the original Internet in 1972 – highly experimental and in need of much real-world testing, iteration, and refinement before it can hope to achieve, like the original internet, far more than it was designed for. Even so, CCNx is a useful case study to analyze network design, and is already demonstrating promise in early experiments.

Before we compare their features more deeply, let's first look in more detail at each network design in terms of its architecture and implementation.

2 Internet Detail

2.1 Architecture

If a good design is nothing more than a diagram of its forces (Thompson 1917), what were the forces that forged the Internet? The top level goal for the DARPA Internet Architecture was to develop an effective technique for multiplexed utilization of existing interconnected networks, yet there were also a number of other design goals, in order of importance:

1. Internet communication must continue despite loss of networks or gateways.
2. The Internet must support multiple types of communications service.
3. The Internet architecture must accommodate a variety of networks.
4. The Internet architecture must permit distributed management of its resources.
5. The Internet architecture must be cost effective.
6. The Internet architecture must permit host attachment with a low level of effort.
7. The resources used in the Internet architecture must be accountable.

These were the primordial forces that provided the first template for Arpanet and TCP/IP (Clark 1988). Impressively, almost every feature is present to some degree in the current Internet, even if they were not present initially. 1 & 2 were there from the start, yet 3 & 4 arguably didn't happen until the switch from original Arpanet to CSNET/NSFNet in the late 80s (behind the expansion being a decentralized routing protocol (BGP) replacing EGP) (Forouzan 2006). As with 5, cost effectiveness arguably wasn't proven until the primary networks were pushed to industry (IBM, Merit, MCI) in the early 1990s (Forouzan 2006). 6 – network access & host connectivity is effortless in an era of smart phones, WiFi, and telecommuting. However, 7 – accountability - not coincidentally the last in the list – remains the most poorly implemented. While accountability was present in the original paper by Cerf and Kahn, a feature that may have made it easier was actually removed early on – that of controlling traffic at the transport layer with both with bytes and (optionally) packets, instead of just with bytes (as is in TCP) (Clark 1988). Yet as with out-of-order packets (discussed below), this missing feature is actually an enhancement, as it ensures all traffic is on equal footing in the transport layer, and is not burdened (nor overwhelmed) by accountability or re-transmission. While it's possible this functionality could be supported w/ today's technology and commercial interest, TCP is too entrenched to have such a re-tooling.

The Internet is undoubtedly a success – but as we see, there were cost/benefit tradeoffs in early implementation that affect us today. Let's look further at what has been implemented since inception.

2.2 Implementation

The evolutionary and incremental development of the Internet emerged not just from the architectural aspects indicated above, but also from active research and experimentation between the architecture and practical reality. Packet switching, the Datagram, End-to-End service & layering – these concepts, combined with research, taken purely architecturally, we arrive at the OSI model (Forouzan 2006).

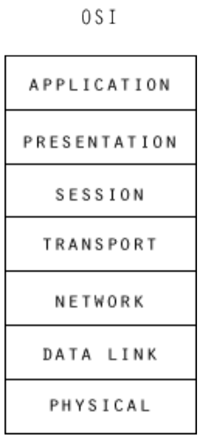

Figure 2-1
(Forouzan 2006)

Each layer has it's own specialized role, and the reasons for this are very nuanced, not even apparent to the early engineers except through experimentation (Clark 1988). For instance, by design, the transport layer has no concept of packet sequence number - packets may thus arrive out of order. Ordered packet sequences are thus constructed in the session layer, not in transport layer. The reason for this is primarily because the overhead for ensuring transmission order was itself often times the cause of failure in early networks (Clark 1988). By having the transport layer explicitly – and only - concerned with routing stateless datagrams around, it can have greater throughput for all packets than having to re-sync or transmit a full stream if a single packet is out of order. The re-transmission is then left up to the session layer & above, and is usually unnecessary (this is why TCP split into distinct TCP & IP in late 70s) (Forouzan 2006). Voice applications, for instance, could have a tiny (typically imperceptible) moment of silence, as by the time the lost packet could round-trip again, the conversation would've moved past that moment anyway. A file transfer application can assemble a file from out of order packets. Thus out of order packets are a feature, not a bug – and leaving such control to the application and out of the transport layer is perhaps counterintuitive at first but experimentally and ultimately most efficient.

As we see, layering is a very nuanced, but critical concept in network design. The OSI model is developed as a guideline to keep layers distinct, from the physical wires up to the application.

While the OSI model is relevant as an architectural guide, the model that was and is actually out in the world is the TCP/IP model (OSI was developed after TCP was already entrenched). The TCP/IP model is now presented along side the OSI model; while there are slight changes to the specific layers, the ordering and concepts remain identical.

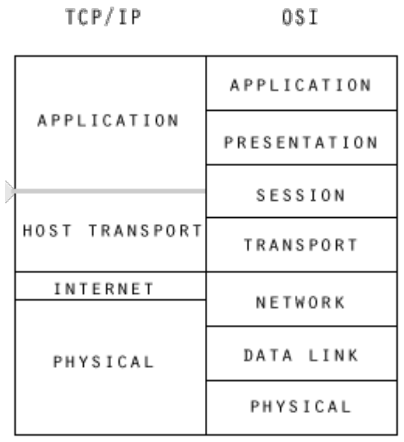

Figure 2-2 (Forouzan 2006)

Note the encapsulation of the application, presentation, and the OSI session layers into the TCP/IP Application layer. Likewise, the TCP/IP transport contains both OSI session and transport layers. The physical TCP/IP layer then includes the OSI network, the link layer, and physical infrastructure. In the TCP/IP layer model, the Internet is the 'thin waist' – that which gets data from user space (application, transport) to the world at large (physical) and back again, as quickly as possible (via established dynamic routing schemes like OSPF) (Forouzan 2006).

As for thin waist – let's look at the Internet today, what has been built with TCP/IP.

2.3 Current State

The versatility of the datagram and packets allowed a wide variety of networks, applications, protocols, and services to grow. HTTP enabled the World Wide Web. SSL/TLS allows secure communications for authentication and financial transactions. SIP/VOIP allows for voice communication & telephony gateways. RTSP allows for real-time video and audio streaming. The sheer flexibility of the layered architecture, combined with aggressively agnostic network layer ('thin waist'), has proved to validate the architecture & design decisions. The World Wide Web being implemented 20 years after Arpanet demonstrates this flexibility; it's but a single protocol at the application, presentation, and session OCI layers (HTTP) – but has resulted in unprecedented economic and social impact (La Rue, 2011). All of this is possible because of the flexibility of IP.

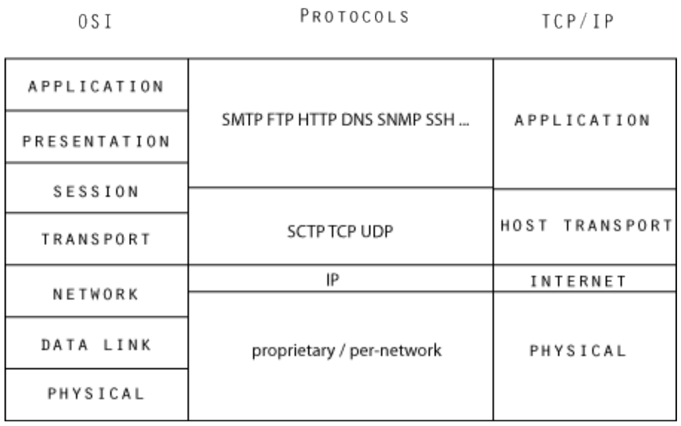

Figure 2-3 (Forouzan 2006)

Now that we have a sense of it's history and architecture, let's now look at what an alternative Internet could look like.

3 CCN Detail

3.1 Architecture

Content-Centric Networking (CCN) was started with the idea of building a new protocol from how people use the existing Internet. "Network use has evolved to be dominated by content distribution and retrieval, while networking technology still speaks only of connections between hosts. Accessing content and services requires mapping from *what* the users care about to the network's *where*" (Jacobsen et al 2009).

Thus CCN has no notion of host at its lowest level. A packet names not location, but content itself. However, the architecture still preserves simplicity of TCP/IP [including store and forward routing & packet switching] to ensure robust scalability (Jacobsen et al 2009).

This data-centrism is furthered by a security model of asymmetric encryption that authenticates data with its publisher at publish-time. By securing content itself, rather than the 'channel' it travels on, CCN can avoid many of the channel/host based vulnerabilities that plague IP networking (like Denial of Service, man-in-middle) (Jacobsen et al 2009). DDOS arguably becomes irrelevant in CCN for at least 3 reasons: content caching mitigates targeted denial of service, content is not forwarded without interests, and multiple interests for same content are collapsed into a single interest ("NSF FIA PI Meeting" 2011).

Another unique feature of CCN is the 'temporal distribution' of packets. Whereas an IP packet belongs to a single point-to-point conversation, and has no value after being forwarded; CCN packets are self-identifying and self-secure such that each packet can be of use to multiple consumers (i.e., YouTube videos, popular news). The savings is considerable when factoring that TCP conversations account for 90 percent of Internet traffic (Jacobsen et al 2009). This new temporal distribution is also spatial, so when the last-hop changes (as in a mobile commuter), only the last link need re-address, and the user can take advantage of the entire upstream cache – whereas on current internet, the entire chain must be re-rerouted from host to client, all existing packets lost due to 'shared fate' of TCP networking (where when either end looses connection, all packets are destroyed) (Floyd et al, 1995).

3.2 Implementation

CCNx is an open source library that essentially emulates the functionality of a CCN router as an overlay on top of IP so it can take advantage of existing networking hardware. There is a CCND daemon, and routing is currently statically controlled (though as we'll examine later, dynamic routing is under development). Even so, at the moment, it is new service/application – similar to HTTP. The critical difference is that CCNx is designed to replace everything above the Link Layer, with technical potential to actually replace TCP/IP (Jacobsen et al 2009).

3.3 Current state

There are C, Java, and Python libraries that allow developers to use CCN. As of writing, we are about 9 months into the first wave of applications written with CCN. As with the original Internet, this is a critical phase where theory meets experiment and forges implementation. Some features of CCN have already been scrapped (bloom filters, a form of query language for interest packets, are being deprecated) (Parc 2011). Others are being introduced, like a new 'sync' api for set reconciliation, while any accidentally asymmetric behavior is being rooted out & fixed PARC, Xerox. (2011). It is of hope that over the next 18 months of research, the library can be stable enough for use beyond research & in production environments.

4 Comparison & Analysis

4.1 Packets

Below we see high-level outlines of TCP & NDN packets. Note that NDN, unlike TCP, does not contain source or destination addresses, and that the 'data' portion of TCP is actually an entirely separate packet in NDN – along with the Interest packet, there are two types of NDN packets to TCP's one. Furthermore, both NDN packets have cryptographic signature, unlike TCP, which relies on higher-level transport & application layers to provide security.

TCP / IP packet format

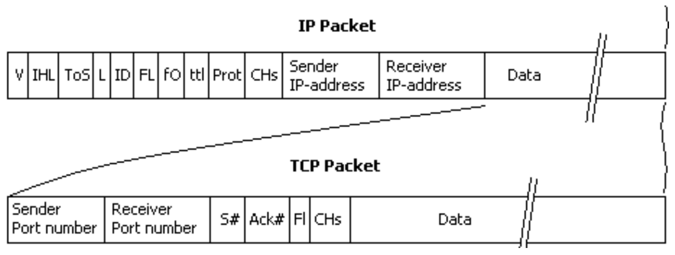

Figure 4-1 (Forouzan 2006)

NDN Interest & Content packet format

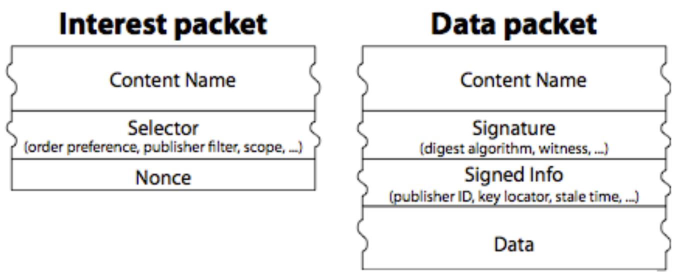
Interest packet
Content Name
Selector
(order preference, publisher filter, scope, ...)
Nonce
Data packet
Content Name
Signature
(digest algorithm, witness, ...)
Signed Info
(publisher ID, key locator, stale time, ...)
Data

Figure 4-2 (Floyd et al)

4.2 Routing and Multicast

TCP/IP Routing overview

For our purposes of architectural comparison, we can ignore BGP, TCMP, and routing intricacies, instead just noting four fundamental details with IP routing.

First, unicast packets must flow on a single interface. When there are multiple ports on a router, data will flow to the port that is closest to the destination (as indicated by the routing table).

Second, data can ‘loop’ – an outgoing packet can become an incoming packet, ad infinitum (or, rather, until TTL counts down), and physical networks must thus be designed to avoid loops.

Third, all data is routed, be it a request or response.

Fourth, we note that IP multicast works by clients sending a unicast join message to a group address (224.X.X.X). Any source can then send packets to the entire group (whether the source is a member or not) by sending them to the group address, which then forwards to the multicast member tree (Forouzan 2006).

NDN Routing comparison

To further understand NDN, let’s look at how it handles routing compared to those four TCP packet routing behaviors.

First, interests can be forwarded on all router interfaces (or by any gradient applied to the ports) – they needn’t just flow from one port. This allows simultaneous ‘broadcast’ to find data, as well as opportunity to weigh the ports per interest namespace; both result in faster and smarter network.

Secondly, it's impossible for CCN packets to loop – each packet has a nonce at the end to determine uniqueness (Jacobsen et al 2009). This feature eases datacenter, ISP and network construction and management.

Third, only interest packets are routed; data packets simply follow the reverse path of the interests.

Fourth, multicast is handled per-node, per interest. For example, in the event a router receives multiple interests for the same upstream content, there will be only one upstream interest expressed; yet the returned content will go to all waiting parties (Jacobsen et al 2009). In this way the equivalent of the IP group address is the content name itself. Note however this is per-packet; there is no permanent 'join' or 'leave' – though such functionality has been implemented at NDN application level for audio conferencing applications (Parc 2011).

Another improvement is that all CCN communication is local. Unlike end-to-end design of TCP, there are no points between sender and receiver that are not involved in flow control, and thus no need for either end to mitigate congestion (Jacobsen et al 2009). Congestion is further reduced with the request/response relationship of an Interest & its corresponding content, it's possible to get a TTL per namespace per port. This allows CCN routers to dynamically construct topologies that are close to optimal for both bandwidth and delay (Jacobsen et al 2009). In this manner, a CCN network is more self-optimizing than TCP/IP.

Lastly, dynamic CCN routing is still in development, and will leverage much of IP routing success that's present in Quagga, including OSPF. "Any routing scheme that works well for IP should also work for CCN, because CCN's forwarding model is a strict superset of the IP model with fewer restrictions (no restriction on multi-source, multi-destination to avoid looping) and the same semantics relevant to routing (hierarchical name aggregation with longest-match lookup)" (Jacobsen et al 2009).

Even so, being in the development stage, the CCN team is also reviewing some promising features that will allow it to scale even more efficiently. One of these features is a 'Hyperbolic

Routing table' – the key idea being that each router can determine its relative co-ordinate in the (necessarily vast) network, and prioritize its data flow across ports so that there is a sense of how 'near' or 'far' the various network ports (and thus data of interest) are from itself (PARC 2011).

4.3 Caching & Performance (HTTP vs CCNx)

When CCNx is compared to lighthttp & squid (HTTP (and thus IP-based) content hosting & caching) the results are about 10 times slower. It's not quite a fair test – CCNx is only at version 0.4 as of writing, and lighthttp & squid have been around for over a decade and are the system of choice for production environments for Wikipedia and Flickr and many others (Squid 2011). However, even with this performance gap (and after compensating for it with adjusting file size per platform) we can see a clear difference in download times when dealing with multiple hosts requesting the same content on the same topology. The CCN curve is not only more flat than the TCP system (with approximately 5 times less variation), speed actually increases for the first 15 clients – something impossible under HTTP even with clever Squid caching.

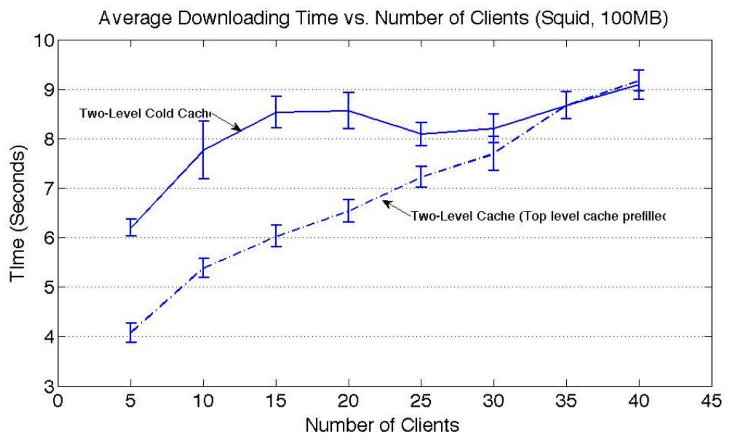

Figure 4-3 (PARC, Xerox 2011)

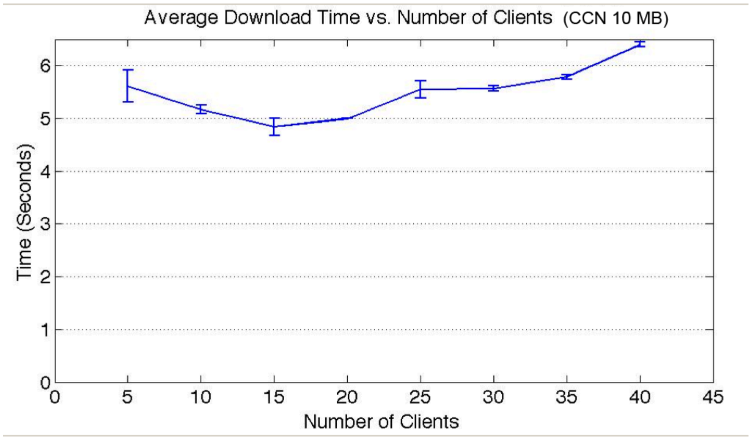

Figure 4-4 (PARC 2011)

4.4 Security & Embedded Control Systems (UDP vs CCNx)

CCN, unlike TCP/IP packets, are cryptographically signed. While this adds computational overhead, it shows promise as an architectural decision. TCP security must be added above network layer (like SSL/TLS) and is always vulnerable to man in the middle and other architectural exploits of TCP itself. CCN security is by default, at all layers of CCN. Each router and application has keys, and all interest and content packets are signed with keys (Jacobsen et al 2009).

However, the way an application chooses to implement security is flexible – it can choose to essentially ignore public signatures (not verifying the content's signature), or verify (proving providence / authority of content), and can even go so far as to use the public asymmetric keys to derive a symmetric 'session' key (via Diffie-Hellman exchange) which is much faster for control systems and real-time applications like multimedia conferencing.

Below we see performance of a single packet round trip – UDP (IP), CCN unsigned, and CCN (asymmetrically) signed & verified, with name/URL of 10, 100, and 1000 bytes. As with the Caching tests, this is likely an unfair comparison as CCN is early, optimized code – and for our purposes a TLS session is arguably a better comparison than raw UDP. Even so, we get a sense of NDN performance on an embedded system.

CCN architecture is especially tuned to embedded systems and Disruption Tolerant Networking – the lack of IP lookup & routing in names can provide power efficiency (once CCN native routers exist), and the routers provide data diffusion / caching (Heidemann et al 2001) 'for free'.

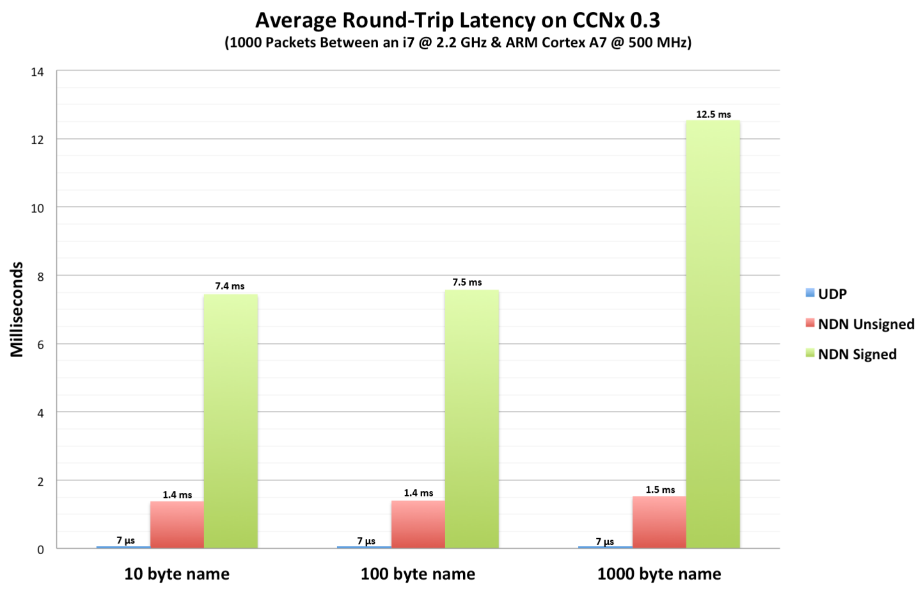

Figure 4-5 (PARC 2011)

5 Conclusion

We've seen how the success of the Internet can be traced to some very early architectural decisions like store & forward, packet switching, the datagram, and 'thin waist' and 'end to end' design of IP. Likewise, the success of CCN will be judged by how well it leverages these developments while building on it's own novel design to bridging the world of host-centrism into content-centric networking. As we saw most notably in the performance results, CCNx is still too young to be compared outright to mature technology such as IP and HTTP in terms of raw performance, but we see it has enough promise to warrant future research & comparison, especially in the realm of security, content distribution, mobile computing, and control systems. Also of note, CCNx has also embraced a cultural development born and raised on the Internet – it's open source software. Anyone that wants to participate in the development of a potential future Internet can do so at ccnx.org.

6 References

Baran, P. (1962). On distributed communications networks. RAND Corporation, Santa Monica CA.

Carofiglio, Giovanna, Massimo Gallo, Luca Muscariello, and Diego Perino. France. Modeling Data

Transfer in Content-Centric Networking. Alcatel-Lucent, Orange Labs, France Telecom, 2011. Web. 1 Jul 2011. <http://www.fp7-pursuit.eu/PursuitWeb/>.

Cerf, V, & Kahn, R. (1974). A protocol for packet network intercommunication. IEEE Journal.

Clark, D. (1988). The design philosophy of the darpa internet protocols. ACM, 8(0-89791-279-9), 106-114.

Forouzan, B. (2006). Tcp/ip protocol suite. New York, NY: McGraw-Hill.

Floyd, S, Jacobson, V, McCanne, S, & Liu, C. (1995). A reliable multicast framework for light-weight sessions and application level framing. SIGCOMM

Heidemann, J, Silva, F, Estrin, D, Ganesan, D, & Intanagoniwiwat, C. (2001). Building efficient wireless sensor networks with low-level naming. ACM Journal.

Jacobson, V, Smetters, D, Thornton, J, Plass, M, & Briggs, N. (2009). Networking named content. Proceedings of the Conext'09

La Rue, F. United Nations, Council on Human Rights. (2011). Report of the special rapporteur on the promotion and protection of the right to freedom of opinion and expression (17th Session Agenda Item 3). New York, NY

"Named Data Networking." NSF FIA PI Meeting. Berkeley CA, Xerox Parc. 2011. Web. <www.named-data.net>.

PARC (2010). faculty bio. Retrieved from http://www.parc.com/about/people/88/van-jacobson.html

PARC, Xerox. (2011). Proceedings of the Parc ndn retreat Palo Alto, CA: http://www.named-data.net/news.html.

Squid Cache. (2007). Wikipedia, squid web deployment. Retrieved from http://www.squid-cache.org/Library/

SEG, C. (Designer). (1996). Telephone switches. [Web]. Retrieved from http://goo.gl/hc1Pp

Thompson, D. (1917). On growth and form. Cambridge, UK: Cambridge University Press.

Victor, J. (Designer). (2002). The strowger switch. [Web]. Retrieved from http://goo.gl/nuFS3